NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

RE-ENGINEERING AND PROTOTYPING A LEGACY SOFTWARE SYSTEM – JANUS VERSION 6.X

by

Julian R. Williams, Jr.
Michael J. Saluto

March 1999

Thesis Advisors: Man-Tak Shing
Valdis Berzins

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE March 1999	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE: RE-ENGINEERING AND PROTOTYPING A LEGACY SOFTWARE SYSTEM – JANUS VERSION 6.X	5. FUNDING NUMBERS
6. AUTHOR(S) Williams, Julian R. Jr., Michael J. Saluto	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

The U.S. Army is working to develop future generations of constructive combat simulation systems that can take advantage of the wide availability of high-end Personnel Computers (PC). As part of this research and development process, the U.S. Army looked to re-engineer a verified and validated legacy combat simulation into a version that can operate on a PC using the industry supported and widely used Windows NT operating system. Janus, with its availability, familiarity, and applicability, will serve as that re-engineering test case. The re-engineered version of Janus will maintain its existing functionality, and include additional functionality to support Operations Other Than War (OOTW) and expanded Combat Service Support (CSS). In its final form, the results of this re-engineering project will produce the Warrior Simulation. Warrior will serve as a basis for future simulations.

This thesis describes the re-engineering activities required to reconstruct the Janus architecture from a legacy software simulation system into one possessing an object-oriented architecture that complies with Department of Defense's (DoD) High Level Architecture (HLA) standard.

This research indicates that procedural legacy simulations can be converted into an objected-oriented architecture that complies with the HLA standards.

14. SUBJECT TERMS Object-oriented Architecture, Prototyping, Forward Engineering, Reverse Engineering , Combat Simulation , Three-Tier Architecture	15. NUMBER OF PAGES 197
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFI-CATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18
298-102

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

RE-ENGINEERING AND PROTOTYPING A LEGACY SOFTWARE SYSTEM – JANUS VERSION 6.X

Julian R. Williams, Jr.
Major, United States Army
B.S., Hampton University, 1988

Michael J. Saluto
Captain, United States Army
B.S., United States Military Academy, 1989

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
March 1999

Authors: Julian R. Williams, Jr.

Michael J. Saluto

Approved by: Man-Tak Shing, Thesis Advisor

Valdis Berzins, Thesis Advisor

Dan Boger, Chairman
Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The U.S. Army is working to develop future generations of constructive combat simulation systems that can take advantage of the wide availability of high-end Personnel Computers (PC). As part of this research and development process, the U.S. Army looked to re-engineer a verified and validated legacy combat simulation into a version that can operate on a PC using the industry supported and widely used Windows NT operating system. Janus, with its availability, familiarity, and applicability, will serve as that re-engineering test case. The re-engineered version of Janus will maintain its existing functionality, and include additional functionality to support Operations Other Than War (OOTW) and expanded Combat Service Support (CSS). In its final form, the results of this re-engineering project will produce the Warrior Simulation. Warrior will serve as a basis for future simulations.

This thesis describes the re-engineering activities required to reconstruct the Janus architecture from a legacy software simulation system into one possessing an object-oriented architecture that complies with Department of Defense's (DoD) High Level Architecture (HLA) standard.

This research indicates that procedural legacy simulations can be converted into an objected-oriented architecture that complies with the HLA standards.

DISCLAIMER

The reader is cautioned that computer programs developed in this research may not have been exercised for all cases of interest. While every effort has been made, within the time available, to ensure that the programs are free of computational and logic errors, they cannot be considered validated. Any application of these programs without additional verification is at risk of the user.

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGEMENTS

Thanks to all my fellow classmates. Without your guidance, assistance, criticisms, and friendship I would have never made it. I owe a special thanks to my family for all their support.

- Michael J. Saluto

My sincerest appreciation and thanks go to my thesis advisor, Professor Man-Tak Shing. He took me under his wing and worked diligent hours helping me to learn and understand numerous software-engineering ideas and concepts. I know that at times I was slow to understand and often asked the same questions repeatedly. My thanks go to him for his sincere patience and support.

I would also like to give special thanks to Professor Valdis Berzins. His knowledge of the field is extraordinary and he provided me with many insightful ideas, guidance, assistance, and most importantly, patience.

But most importantly, I want to send a very special thanks to my family and friends for their sincere love, confidence, and support. Thanks.

- Julian R. Williams, Jr.

I. INTRODUCTION

A. BACKGROUND

Up until just a few years ago, software development was conducted under the auspices that a system[1] would be developed and maintained for some period of time and then eventually replaced by a totally new future system. However as budget purse strings tighten, the increased dependence on commercial technology and the use of Commercial-off-the-shelf (COTS) products, coupled with the wide spread use of legacy systems, it is becoming increasingly more important and highly economical for concerned agencies to explore re-engineering opportunities and strategies. Legacy systems like Janus embody substantial institutional knowledge which include both basic and refined requirements, design decisions, and invaluable advice and suggestions from military leaders that has been implemented over the years. To effectively use these assets, it is important to employ a systematic strategy for continued evolution of the current system to meet the ever-changing mission, technology and user needs. However, knowledge is difficult to recover after many years of operation, evolution, and personnel change. Janus software was originally written some twenty years ago using, what many now view as, an archaic and ad-hoc methodology. In addition, Janus has experienced a number of updates and maintenance revisions over recent years. Major changes occurred during the transition to version 4.0 with the integration of new terrain features such as roads, buildings, vegetation, water and other man-made features. The end result is a legacy system that lacks the ability to evolve to meet the ever-changing demands needed to continue

[1] The systems discussed in the thesis are software intensive: software constitutes a significant portion of the application. Unless otherwise noted, the term "system" implies "software system".

as a valuable military training and analysis tool.

Re-engineering has frequently been proven to be more cost effective than new development and is also known to better promote continuous software evolution. It is essentially building a new system using the existing system as the basis for requirements or design. Software re-engineering can defined as the systematic transformation of an existing system into a new form to realize quality improvements in operation, system capability, functionality, performance, or evolvability at a lower cost, schedule, or risk to the customer [Ref. 1]. Such improvements often take the form of increased or enhanced functionality, better maintainability, configurability, reusability, and/or other software engineering goals. This process involves recovering existing software artifacts from the system and then re-organizing them as a basis for future evolution of the system.

When re-engineering a legacy system, the use of object-oriented techniques introduces certain complexities into the software analysis process. Primarily we find, as was the case with Janus, that the software system was not originally designed and implemented using an object-oriented approach. Thus, the products of reverse engineering, such as requirements or design specifications, will probably reflect a functionally based approach. As a result, some form of "transformation" of analysis and design is necessary in order to use the specifications.

Once a realizable specification is obtained, it is often very difficult to quickly determine if the specification is in fact a true representation of the desired requirements. Prototyping provides a means to validate system requirements while simultaneously allowing a prospective user with the opportunity to get a brief feel for aspects of the proposed system. It is a well-established approach that can be highly effective in increasing software quality [Ref. 2]. Used in conjunction with conducting a major re-engineering effort, prototyping can be extremely

useful in assisting in many areas of software modification, validation, risk reduction, and the refinement of user requirements.

B. MOTIVATION

In September 1996, USD(A&T) signed a memorandum promulgating the High Level Architecture (HLA) as the architecture of choice for Department of Defense (DoD) simulations. DoD additionally stated that all agencies shall cease further development or modification of all simulations which have not achieved, or are not in the process of achieving, HLA-compliance by the first day of FY 1999, and shall retire any non-compliant simulations by the first day of FY 2001 [Ref. 3]. In April 1998, USD(A&T) reaffirmed that policy stated in the September 1996 memorandum. To recap the full benefits of simulation interoperability and reuse in the near-term, it is important to quickly transition our legacy to the HLA [Ref. 4]. In support of these policies and a vision to improve solider training opportunities, the U.S. Army Training and Doctrine Command (TRADOC) Analysis Center in Monterey, California, (TRAC-MTRY) began efforts to research and develop an HLA compliant, personal computer (PC)-based, high-resolution, multi-sided, constructive combat simulation. The simulation selected for development was Janus. The overriding goal of the project was a prototyped HLA-compliant Janus, coded in C++, operating on a PC platform using the Windows NT (WinNT) operating system [Ref. 5]. Large constructive software simulation systems like Janus typically reside on bulky hardware platforms, usually running a UNIX-based operating system. However, the significantly increased computing power of today's PCs, combined with their low cost, availability, and wide familiarity make them a very attractive and logical choice for future simulation systems.

A major goal in the ongoing research effort in the Software Engineering Laboratory at the Naval Postgraduate School is the construction of a highly automated software engineering environment to support computer-aided design, development, and reuse of large software systems. A number of models and tools have been developed as a result of the research, many of which largely support the task of re-engineering large software systems like Janus. TRAC-MTRY tasked the Software Engineering Laboratory at the Naval Postgraduate School to assist in its effort to re-engineer Janus into an object-oriented system. This thesis describes our efforts to modernize Janus into more maintainable, evolvable system, and one that exploits the recent speed, memory, and power enhancements of today's modern PCs.

C. OBJECTIVES

The primary objectives of our work were twofold; 1) propose a methodology to re-engineer a legacy software system, and 2) produce an executable prototype of the design using the Computer Aided Prototyping System (CAPS) and its supporting specification language, the Prototyping System Description Language (PSDL). The work involved moving Janus, a legacy combat simulation system, from an HP-UNIX based platform, written mainly in the procedurally structured FORTRAN 77 programming language, to an object-oriented programming environment running on a PC platform. It was additionally a goal of the developers to rewrite Janus in an object-oriented programming language, preferably C++. However, prior to rewriting Janus code, the developers needed a completed software architecture of the existing Janus code functionality. The decision to develop an object-oriented architectural design facilitated rewriting Janus in C++ and integration of a Graphical User Interface (GUI). As any experienced software engineer will tell you, a well-designed

architecture is the first step to successfully re-engineering a software system because it acts as a blueprint when designing the desired class structures, objects, attributes, interactions, and needed parameters.

D. THESIS ORGANIZATION

Chapter II provides an overview of the Janus simulation system, including background information on the system's creation and its development. It provides a brief description of the model along with an analysis of the system to include a discussion of the system's strengths and weaknesses.

Chapter III begins with a brief description of the re-engineering process and provides a detailed account into the underlying steps of System Understanding, Reverse Engineering, Application Restructuring, and Forward Engineering. The next section, Building the Object Model, discusses the selected architecture and describes how the model was actually constructed using simulation objects and the event handlers. The chapter concludes with a brief analysis of the process.

Chapter IV describes the procedures used in constructing the executable prototype. It outlines the purpose for the prototype, describes how it was done and then provides some unique insight of lessons learned during the process.

Chapter V summaries the key elements of our re-engineering effort and provides some insightful recommendations for future work and research in the area of legacy system re-engineering.

THIS PAGE INTENTIONALLY LEFT BLANK

II. JANUS OVERVIEW

A. BACKGROUND

The Army used constructive combat simulation models for training since the late 1970s. One of the earliest such models, the McClintic Theater Model, was developed by an Army War College employee by the name of Fred McClintic. The McClintic Theater Model, which quickly became known simply as "MTM", was a prototype for "theater style" constructive simulations used for training, typically at Division level or higher. [Ref. 6]

Janus began development as a contemporary of MTM, but was intended to meet utterly different requirements. Responding to a Department of Energy requirement, Janus was developed as a nuclear effects modeling tool by Lawrence Livermore National Laboratory (LLNL). Fielded in 1978, the Janus simulation was named after the two-faced Roman god of portals who guarded the gates of Rome by looking two ways at the same time. Later, the U.S. Army Training and Doctrine Command (TRADOC) Analysis Center, White Sands Missile Range, New Mexico (TRAC-WSMR) acquired the prototype from LLNL as result of the Janus Acquisition and Development Project. DoD's interest in Janus prompted the development of several parallel versions of the model during the late 1980s and early 1990s. The original version, developed at LLNL, is known as Janus(L) while the version adopted and successfully modified by TRAC-WSMR to meet Army combat development needs is known as Janus(T). Both of these models achieved great success and popularity amongst its users, which promulgated the Army to task TRAC-WSMR to develop a multipurpose system from the best of Janus(L) and Janus(T). The simulation, originally referred to as Janus (A), is now simply referred to as Janus. Janus has gained

immense popularity over the course of its lifetime and is currently in use by several allied nations abroad to include Germany, France, Canada, and Australia. Here in the United States, Janus is used not only by the U.S. Army, but also by the U.S. Marine Corps and the Rand Corporation. The most recent version of Janus, version 6.88, is the analytical model which is managed and maintained by TRAC-WSMR. There is also training version of Janus, version 7.0, currently managed by the Simulation, Training and Instrumentation Command (STRICOM) located in Orlando, Florida. [Ref. 7]

B. DESCRIPTION

Janus is a high resolution, software-based, constructive combat simulation model used by Army leaders at brigade through platoon-sized units as an effective tool for training staffs and analyzing combat tactics. The model simulates battle between two to six opposing forces, depicting actions from individual systems and company-sized units, on up through brigade and regimental-sized units. Each element is viewed on the Janus screen as an icon. The icon may represent one or more pieces of equipment. For example, one icon may represent one tank, or it may represent a platoon of tanks.

Janus is an interactive, closed, stochastic, ground combat simulation that features precise color graphics. It is "interactive" in that command and control functions, entered by military analysts, determine what actions take place in high tempo situations during simulated combat. "Closed" in that the disposition of opposing forces is largely unknown to the players in control of the other force. It is "stochastic" in the way the system determines the results of actions like direct fire engagements, according to the laws of probability and chance. The term ground combat implies that the principal focus is ground maneuver and artillery units. However, Janus also models many other characteristics such

as weather and its effects, day and night visibility, engineer support, minefield employment and breaching, rotary and fixed wing aircraft, resupply, and a highly robust and realistic chemical environment. [Ref. 8]

The Janus system was originally designed to run on the Digital Equipment Corporation VAX suite of computers running the Virtual Memory System (VMS) operating system. The current "open systems" version runs on Hewlett-Packard workstations under HP-UX and supports both Tektronix and X-window workstations. Our work was solely concerned with the HP-UX version using X-windows workstations. The graphics environment, controlled by FORTRAN subroutines, send the appropriate graphics messages through the RTX X-window driver program to an X-window workstation. These subroutines are included in the Janus executables. [Ref. 8]

C. ANALYSIS

Initially written in 1978, the model consists of sixteen executable programs written in FORTRAN 77, with one additional subroutine written in the C programming language. FORTRAN 77 and C are procedure-oriented languages; that is, they are defined in a way that closely models the functions to be automated rather than the computer on which they operate. Janus code development occurred over a number of years, involving many different programmers, using several operating systems and various FORTRAN 77 and C language compilers. In concert with the style of programming used at that time, its data representation, instruction format, pointer usage, and control instructions are highly characteristic of the FORTRAN 77 and C language programming styles. Each program is made up from a number of subroutines and utility functions that were compiled separately and then linked into system libraries. The executable code was then generated from those

system libraries. Several programs share common subroutines.

1. Strengths

As with all legacy systems, Janus has evolved over a number of years and as such, embodies substantial organizational knowledge. Being referred to as a legacy system in itself implies a system that contains many properties worth preserving. Janus has been deployed for over twenty years and has undergone the scrutiny of real users with respect to its functionality meeting their real needs. More specifically, the Janus simulation system uses several very unique algorithms that have been very carefully constructed and refined over the life of the system. These algorithms are used to simulate very highly complicated state environments to include weather and temperature effects, chemical reactions, cloud movements, direct and indirect fire events, and probabilities for line-of-sight, target identification, recognition, and acquisition.

In addition to the extraordinary algorithms used in Janus, the simulation maintains a huge database, which describes weapon systems extensively and in detail. Individual fighting systems have distinct properties such as dimensions, weight, carrying capacity, speed, weapons and weapons capabilities like range, type of ordinance and ammunition basic load. [Ref. 9]

2. Weaknesses

Many of the Janus system variables and parameters used in the code are passed between the various programs explicitly or via the use of global data sets, often called FORTRAN common blocks. This form of programming uses a "structured" design strategy and may be considered highly functionally oriented. Function-oriented design is

characterized by decomposing a system into a set of interacting functions that all share a common centralized system state. Algorithm details and parameter manipulations are often hidden in this style of design whereas the system state information can be viewed and shared by all. This sharing of system state information can create serious problems since a function can change the system state in a way that is inconsistent with the expectations of a subsequent function. Furthermore, changes to a function and the manner in which that function uses the system state information may also create many unanticipated changes in the behavior of other functions.

As previously mentioned above, the Janus simulation system was originally designed using a function-oriented approach to software development. In this approach the design is commonly decomposed into a set of interacting units where each unit has a clearly defined function. Design components in this approach are normally highly cohesive around functions that operate on the global data sets, which make modifications and upgrades very difficult and time consuming. For instance, the Janus database contains a number of weapon systems to include tanks, helicopters and many other combat weapon systems. Adding a new tank with characteristics dissimilar from those of other tanks in the current database would require an immense amount of work and the programmer would have to make modifications in many different sections of the software. Changing the implementation of a weapon system or adding a new service, in almost any scenario, would require the programmer to modify numerous sections of the software. Furthermore, many of the newly modified sections would seem to many, to be totally unrelated to the originally desired implementation change or service addition.

Finally, although FORTRAN is probably the language best suited for mathematical and scientific programming, it possesses limited program structuring facilities and has very limited support for data structuring. Thus, one can easily see that the current version of Janus is not easily modified and maintainability is often very difficult and time consuming.

III. METHODOLOGY

A. THE RE-ENGINEERING PROCESS

Software re-engineering combines forward and reverse engineering techniques to make a system that is more maintainable and more evolvable. Figure 1 illustrates this process. Software re-engineering may be defined as any activity that improves one's understanding of a software system and/or improves the software itself [Ref. 10]. This definition partitions software re-engineering into two components: software understanding and evolution. The first component, software understanding, involves those activities that support program comprehension, such as measurement, browsing, system understanding, and reverse engineering. The second component, evolution, includes those activities geared toward software evolution such as redocumentation, restructuring, and forward engineering. The approach taken in our research followed a sequence of system understanding, reverse engineering, then followed by application restructuring and finally forward engineering. The details of each activity are described in the following paragraphs.

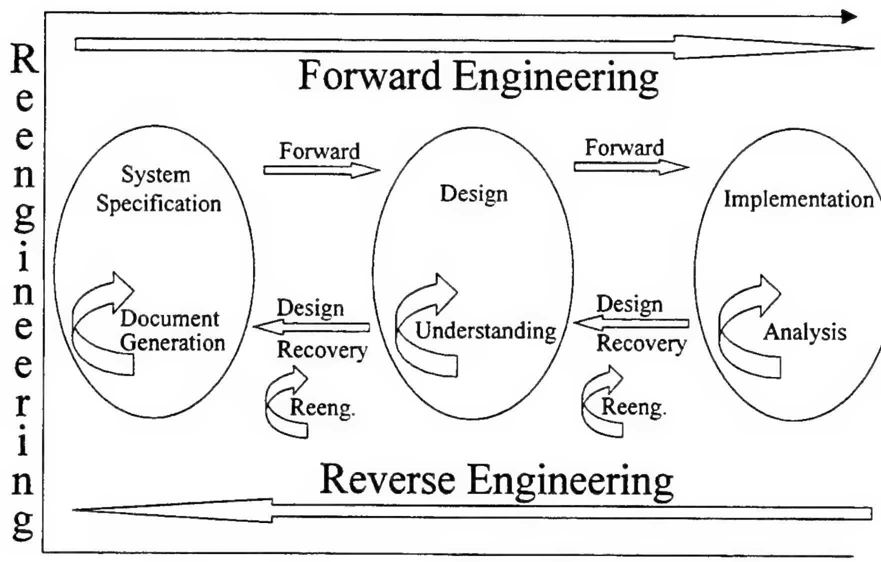


Figure 1. Re-engineering Process

1. System Understanding

System understanding reflects the process of creating and maintaining an understanding of the system through analysis, elicitation, and capture [Ref. 11]. The first step in our process took the form of a series of brief meetings with the client, TRAC-Monterey, which also included a short demonstration of the current software system. We asked questions and made several notes on the system's operation and it's current functionality. We paid particular attention to the client's view of the system to gather their ideas on its strengths, weaknesses, and desired and undesired functionality. Additionally we collected copies of the Janus User's Tutorial manual, Janus User Manual, the Software Design Manual from a previous version of Janus (3.X/UNIX), and the Janus Version 6.88 Release Notes. Our goal was to gather as much information as we could about the current existing system to aid in gaining a clearer understanding of its present functionality. This provided a sound foundation upon which to move up one

level of abstraction from implementation and focus on the system's underlying architectural design. The intent of this procedure was to ensure that the systems' current functionality was not lost nor misrepresented in the transformation into a more abstract, modular format.

2. Reverse Engineering

Reverse engineering may be defined as the process of analyzing an existing system; identifying its system components, abstractions and interrelationships; and then creating the respective representation [Ref. 11]. In similar fashion, the focus of this step was to abstractly capture the system's functionality and then produce modules that would most accurately represent that functionality. Armed with the Janus source code, we proceeded to divide the code by directories amongst each team member. Each team member was assigned roughly six to seven directories to explore, examine, and gather information. Using strictly manual techniques with UNIX commands and review procedures, we were able to get a fairly good idea of what each subroutine was designed to do. We additionally used the Software Programmers' Manual to aid in better understanding each subroutine's function, its required parameters and coordinating subroutines. In doing so we were able to group the subroutines by functionality to get a better understanding of the major data flows between programs. Using that knowledge, we developed functional models from the data flows. We used an automated tool known as CAPS, developed by Professor Luqi and the Software Engineering group at the Naval Postgraduate School, to assist in developing the abstract models [Ref. 12, 13]. CAPS allowed us to rapidly graph the gathered data and transform it into a more readable and usable format. Additionally, CAPS enabled us to develop our diagrams separately with

the associated information flows and stream definitions, and then join them together still under the CAPS environment. These diagrams were then used to generate an executable model of the system's architecture.

3. Application Restructuring

Next, we proceeded to develop the object models of the Janus system using the aforementioned materials and products to create the modules and associations amongst them. Chikofsky defines restructuring as the transformation of representations at the same level of abstraction while preserving the system's external behavior [Ref. 11]. The transformation of the functional models to object models was probably the most difficult and most important step, primarily because it required a great deal of detailed technical analysis and astute focus to mentally transform the current chaotic assembly of data and functions into small, realizable objects each with its own set of attributes and operations. In performing this step, we used our knowledge of object-oriented analysis applying concepts from Object Modeling Technique (OMT) and the Unified Modeling Language (UML) notations to create the classes and associated attributes and operations. This was really a crucial step because we had to ensure that the classes we created accurately represented the functions and procedures currently found in the original system. We used the HP-UNIX systems at the TRAC-MTRY facility to run the Janus simulation software to aid in verifying and/or supplementing the information we obtained from reviewing the source code and documentation. This step enabled us to better analyze the simulation system, gaining insight into its functionality and further concentrate on module definition and refinement. Our goal was to develop viable class representations from the gathered information, which would best represent the software's current functionality.

4. Forward Engineering

As in conventional software development, forward engineering begins with the system specification. The newly developed object models along with our knowledge gained during the system understanding phase served as our system specification for the realization of the new system.

During this phase of the re-engineering effort, our team focus was to move forward to develop the desired system for the Janus Simulation System from our newly created object models. Again we closely analyzed each object, its associated attributes, and operations to ensure we captured an accurate representation for future development and integration into our newly planned object-oriented architecture. The team met several times each week for a period of nearly three months to discuss the details of the object models and the proposed structure of the Janus object-oriented architecture. We also presented our findings weekly to the Janus domain experts at TRAC-MTRY and members of Rolands & Associates, a local software developer also involved in the project. Our re-engineering team additionally presented findings to numerous other technical organizations within the software simulation development community to include the OneSAF project, the Combat21 project, and the National Simulation Center at Fort Leavenworth, Kansas. The gathered feedback from each session was then reviewed and analyzed for possible incorporation into the object models for the Janus core elements and development of the object-oriented software architecture.

B. BUILDING THE OBJECT MODEL

1. The Three-Tier Object-Oriented Architecture

We observed that possessing a solid software architecture is one of the key elements in successfully re-engineering a legacy software system. Realizing this, we opted to use a layering concept to provide an application architecture that would minimize the negative impacts due to change in the base technology or in the addition of any new application features. Layering application architecture has also been shown to enhance the concept of software reuse by creating additional layers to separate the user system interface client environment from the database management server environment and other services. Some systems have achieved upwards of 60% reuse of software within one domain of an application. [Ref. 14]

In designing our system, we wanted to use an object-oriented system architecture to serve as the basis for our reengineering effort. Our current top-level communications structure of the existing software did not decompose the system into clearly defined subsystems where each subsystem comprised a related set of functions sharing a common purpose and having a well-defined interface.

To create a truly object-oriented system architecture we chose to decompose the Janus system vertically into layers. Each layer was built in terms of the ones below it. The lower layer provides the basis for implementation to the layers above it. Thus, a subsystem can communicate to a lower layer via a client-server relationship where clients must know the interface of the server to use its provided functionality. This type of architecture highly promotes system flexibility and reuse by allowing a subsystem or

layer to be modified, rewritten, and /or replaced without affecting or disrupting the operation of the entire system.

After deciding upon a layered software architecture, we additionally selected to use a closed architecture. This meant that each layer was limited to communicate only to the layer immediately beneath it. Implementing this limitation reduced the dependencies to just two layers: the current layer and the layer immediately beneath it. Reducing dependencies amongst these layers of the architecture worked to localize changes to just one layer as long as all of the interfaces associated with that layer remained the same. Subsequently, if changes to an interface must occur, this type of architecture limits those changes to the system to just the two layers interacting via the newly changed interface.

In our design, we chose to use a classic three-tier architecture (Appendix A) to serve as our basis for the Janus system. This architecture took into account all the benefits listed above by providing three layers of representation: a Presentation layer, an Applications layer, and a Storage/Network layer. We mapped the functionality represented in the current Janus top-level communications structure to our three-tier architecture. The Janus User Interface maps directly to the presentation layer, which is relatively free of application processing. The Applications layer contains the meat of the system to include the core elements that perform the major event-processing tasks in Janus, and the other components required for system operation. We additionally chose to further divide the Applications layer into two sub-layers (Appendix A): a Domain layer, which provides services to the Janus User Interface and a Services layer to provide communication and access to the storage devices and the network. This finer decomposition of the middle layer promptly introduced the concept of a multi-tiered

architecture as opposed to three, however the idea of a single "middle tier" remains. The additional layers allowed us to further separate the responsibilities imposed by the three-tier architecture and develop more modular, specialized, reusable components. We were then able to map most of the functionality to the Domain layer, thus providing services similar to those currently provided by the Janus GUI. Most of the work is done in the Domain sub-layer. The components here perform the majority of the system processing functions and application execution. The Services sub-layer provides access to both the Janus Database and DIS/HLA infrastructure.

Using a three-tier architecture provided a clearly defined interface between each layer, which allowed changes to a particular layer to be localized to that layer. For instance, if the Janus database had to be completely changed due to size and format constraints, this type of architecture would allow changes to be localized to just the storage layer as long as the interface between the storage and services layers did not change. In a similar fashion, if for instance the interface between database utilities and the database did not provide the proper services, a change to that interface would be needed to include any new interface requirements. This newly changed interface would then demand appropriate changes be made to the services and storage layers to accommodate those new interface changes. However, nothing in the domain layer would need to be changed.

2. Building the Simulation Objects

After developing the three-tier architecture, our next goal was to expand this architecture to encompass the functionality of the current Janus. As part of our reverse engineering effort, we focused our attention on the Janus Combat Simulation and Core

Elements. In the current version of Janus, a user can create a Scenario by choosing run parameters that control the environment, which consists of a terrain and weather conditions databases. The user can also establish combat forces. Each force would then contain a collection of combat units, command and control graphics, and any selected obstacles.

Since Janus is such a large system, our first step was to build small, coherent, realizable objects, each with its own attributes and operations, that would accurately represent the functions and procedures of the current software. In order to accomplish this task we divided the components of the current Scenario amongst the team members. Each member reviewed the source code and documentation and created objects with associated attributes and methods. After each member had an opportunity to work on his or her object, the re-engineering team met to discuss the object models for the Janus core elements and the object oriented architecture for the Janus system. As each member presented their finding, others provided feedback. To ensure greater accuracy, we presented our findings to the Janus domain experts from TRAC-MTRY and Rolands & Associates. Based on their feedback the team revised the object models.

We focused primarily on the attributes of the system to create the objects since the attributes were fairly well defined in the Janus documentation. By tracking the attributes, we were able to find insights to how Janus manipulates data structures. Data structures in Janus are FORTRAN arrays. Operational parameters are spread across a number of arrays and are identified by an array index value. We found that information regarding a particular object was not encapsulated. Encapsulation and information hiding allow the programmer to change the internals without affecting the user provided the interface does

not changed. By encapsulating the attributes within the objects, we greatly reduced the sophisticated programming skills required to understand Janus' data representation and also reduced the risk and cost to modify future systems.

As part of revising our object models, we often tried to structure our objects to obtain the greatest benefits of Object Oriented Programming (OOP) (Appendix E). Our first effort was to arrange the objects into hierarchies. Hierarchies provide many valuable advantages such as type extensions, inheritance, and dynamic dispatching. A specialized object would inherit the primitive attributes and methods of its parent, but maintain the flexibility to add or override attributes and methods in order to provide specific behaviors to the object. For an example, if there was a new tank that exhibited slightly different movement behavior than other tanks, a designer could create a subclass of the current Tank Class. The subclass would then inherit all the attributes and methods of its parent class. The designer could take advantage of similar behaviors by deciding not to override any of the common methods and attributes of the Tank Class. He could also add specific behaviors by overriding others or by adding additional methods and/or attributes.

3. Building the Event Handler Objects

After creating base objects with their associated attributes and simple methods, we needed to ensure the primary functionality of the Janus Simulation System was properly captured in our model. In order to accomplish this task, we looked at the existing Janus code architecture. Central to the Janus Combat Simulation Subsystem is the program RUNJAN, which is the main event scheduler for the simulation. RUNJAN determines the next scheduled event (called a "process" in the Janus User Manual) and executes that event. The existing Janus Simulation System uses 17 different categories to

characterize the events. RUNJAN then handles these 17 events using the following event handlers:

1) DOPLAN - Interactive Command and Control activities
2) MOVEMENT - Update units positions
3) DOCLOUD - Create and update smoke and dust clouds
4) STATEWT - Periodic activity to write unit status to disk
5) RELOAD - Plan and execute the direct fire events
6) INTACT - Update the graphics displays
7) CNTRBAT - Detect artillery fires
8) SEARCH - Update target acquisitions, choose weapons against potential targets, and schedule potential direct fire events
9) DOCHEM - Create chemical clouds and transition units to different chemical states
10) FIRING - Evaluate direct fire round impacting and execute an indirect fire mission
11) IMPACT - Evaluate and update the results of an indirect round impacting
12) RADAR - Update an air defense radar state and schedule a direct fire event for "normal" radar
13) COPTER - Update a helicopter states
14) DOARTY - Schedule an indirect fire mission
15) DOHEAT - Update units' heat status
16) DOCKPT - Activity to perform automatic checkpoints
17) ENDJAN - Housekeeping activity to end the simulation

The existing event scheduler relies on global arrays and matrices to maintain attributes of the objects. Since our first task was to move these attributes to the corresponding objects, our second task would be to distribute the event handling functions to the individual objects. However, as discovered from our first task, many of the current event handler categories contained redundant code and did not seem to be very coherent to the objects we created. Thus, we realized the need to redefine some of the event categories in order to provide a uniform framework and to eliminate redundant coding of similar or identical functions. This also allowed us to take advantage of the dynamic dispatching capabilities of the event handling functions inherit within our object-oriented architecture. For example, the set of event handlers used for a particular unit to search for targets, select weapons, prepare for a direct fire engagement, and then execute that direct fire engagement differs depending upon whether the unit has a normal radar, special radar, or no radar at all. The existing Janus Simulation System uses the RADAR event handler to carry out the entire procedure if the unit has normal radar. However, it uses the SEARCH, RADAR, and RELOAD event handlers to carry out the procedure if the unit has special radar. Finally the system uses the SEARCH and RELOAD event handlers to conduct the procedure if the unit has no radar at all.

Upon analyzing the Janus Simulation System event handlers, we were able to successfully reduce the total number of event handlers needed in the simulation, from 17 to 14, by eliminating identified redundant code. Our 14 event handlers are as follows:

1) DOPLAN - Interactive Command and Control activities
2) MOVE_UPDATE_OBJ – Moves and update all objects in the simulation
3) SEARCH – Based on all detection devices, searches for potential targets

4) CHOOSE_DIRECT_FIRE_TARGETS – Once search is complete chooses best target to engage. In future simulations, implementations may allow users to choose targets.
5) COUNTERBATTERY – Simulates counter battery radar to find potential targets
6) DO_DIRECT_FIRE – Executes direct fire events and updates ammunition status
7) DO_INDIRECT_FIRE – Executes indirect fire events and updates ammunition status
8) IMPACT_EFFECTS – Calculates results of round impacting.
9) UPDATE_HEAT_STATUS – Updates units' heat status.
10) UPDATE_CHEMICAL_STATUS – Update unit's chemical status
11) DISPLAY – Updates the graphics display.
12) WRITE_STATUS – Periodic activity to write units status to disk.
13) CHECK_POINT – Activity to perform automatic checkpoints
14) END_SIMULATION – Activity to end the simulation.

As can be seen in Appendix B, we renamed some of the event handlers to possess more descriptive, meaningful names. We additionally combined some event handlers having similar functionality into ones that were more easily understood and applicable to the actual function. Every event now has an associated simulation object. This associated object is the target of the event. Depending on the subclass to which an event object belongs, the "execute" method of the event will invoke the corresponding event handler of the associated simulation object (Appendix C). The simulation object

superclass defines the interface of the event handlers for the event groups. At the highest level, it provides an empty body as the default implementation for the event handlers. Events are dispatched to the appropriate subclass. The event handler of the subclass overrides the inherited method in order to perform the desired behavior, if there is something more specific that needs to be done for instances of the subclass.

The architecture described above enables a very simple realization of the main simulation loop (Appendix G, Section 25). The pseudo code for the event control loop is as follows:

```
initialization;
While not_empty(event_queue) loop
        e := remove_event(event_queue);
        e.execute();
End loop;
finalization;
```

Note that this same code is used to handle all of the event handlers, including those for future extensions that have not yet been designed. Event objects with associated simulation objects are created and inserted into the event queue by the initialization procedure, the constructors of an object, and the actions of other event handlers. Depending on the actual event, it is inserted into an event priority queue based on time and priority.

Using the old event handlers under current the Janus simulation system to move a tank, smoke cloud, or helicopter required three distinctly different event handlers, MOVEMENT, COPTER and DOCLOUDS respectively. During our analysis we were able to consolidate these three event handlers into one namely, MOVE_UPDATE_OBJ. We observed that although a tank, a smoke cloud, and a helicopter are all distinctly different, each one as an object has the capability to move and update its present location

and other parameters. Thus, the name MOVE_UPDATE_OBJ accurately described the event. Additionally, under the old event handlers, depending on the particular simulation object, the program would have to analyze each object using several conditionals in order to determine which event handler to invoke. Under our architecture, we take advantage of the dynamic dispatching capabilities provided by an object-oriented programming language, to automatically dispatch the event to the appropriate event handler. Thus to move a smoke cloud object, the new architecture will invoke the MOVE_UPDATE_OBJ method of the Cloud Class. This allows the simulation to correctly move the cloud in accordance with the current environment and also update the cloud's size and intensity. Similarly, to move a tank or helicopter object, the new architecture would dispatch to the appropriate MOVE_UPDATE_OBJ method of the Vehicle Class. This would subsequently move and update the object's fuel consumption and other required parameters based on the particular vehicular type.

Our newly designed architecture eliminates the need for the simulation loop to know what kind of object it is handling. Thus when adding an object type not yet designed, the simulation loop does not require additional code to invoke the new object's event handlers. This eliminates the possibility of introducing errors into the existing parts of the simulation by localizing all changes to the newly added object class.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. PROTOTYPE: WARRIOR VERSION 1.4

A. BACKGROUND

Prototyping has increasingly become a widely used methodology to improve the design of software projects [Ref. 15]. A prototype is an executable model of a proposed software system that accurately reflects chosen aspects of the system, such as display formats, the values computed or response times [Ref. 2]. The prototyping methodology is based on an iterative guess/check/modification cycle that relies on prototype demonstrations and customer reactions to validate behavior of the final system. The following diagram in Figure 2 shows how to validate requirements using a prototype process [Ref. 16]:

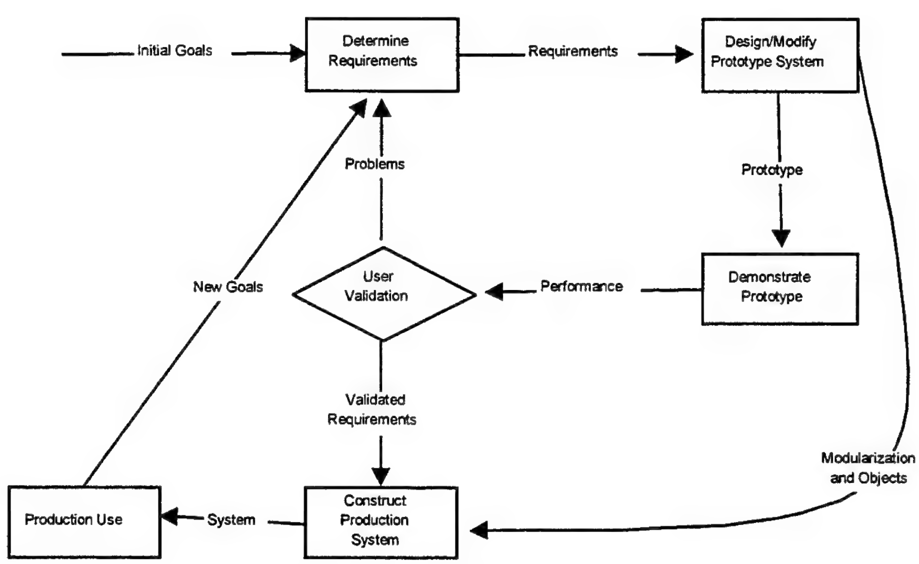


Figure 2. The Prototype Process

Prototyping may be defined as an approach to software development that uses prototypes or executable models to aid both developers and customers alike in visualizing the proposed system, and to predetermine its system properties using an iterative process

[Ref. 2]. The primary purpose of a prototype is to serve as an executable model of selected aspects of a proposed system and to help designers confirm and refine requirements for a software system. Eliminating incorrect or poorly defined requirements early in the software design cycle helps reduce the time and total cost of the system. If these design errors are left to propagate, they may result in a large amount of wasted effort spent developing software to meet incorrect or inappropriate specifications. Much of the time, effort, and cost to produce the product will be thrown out. Designers will have to go back to the drawing board to correctly re-define the requirements.

Computer-aided prototyping serves as a method to automate the design process. Automation allows designers to quickly develop prototypes to analyze software systems. One system currently under research at the Naval Postgraduate School is CAPS. The main components of CAPS (Figure 3) are the Editors, the Execution Support, the Software Base, and the Project Control [Ref. 17].

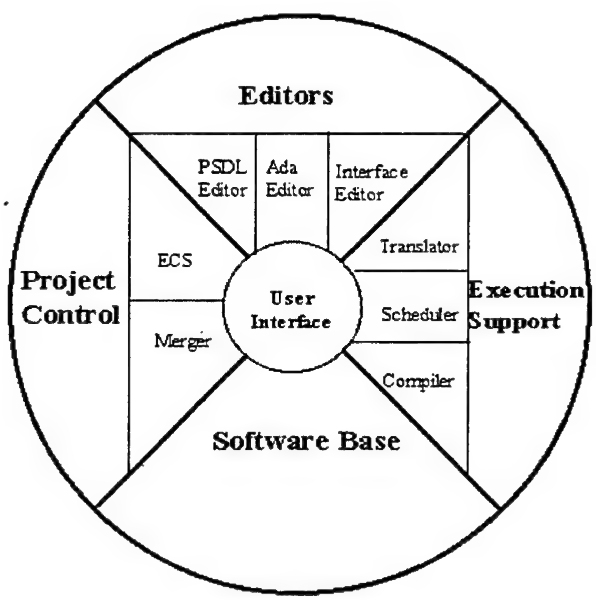


Figure 3. CAPS General Structure

Using the Editors, the designer can sketch system data flow diagrams augmented with timing and control constraints. PSDL is a high-level prototyping language designed to support the specifications of real-time software systems. PSDL also helps organize and retrieve reusable components from the Software Base. The Software Base is a database system that consists of reusable designs and software components. The Execution Support system contains the translator, the static scheduler, the dynamic scheduler, and the debugger [Ref. 18].

CAPS provides many benefits, the most significant of which is that it can automatically generate ADA source code resulting in more reliable systems at reduced production costs. CAPS supports evolutionary prototyping that provides the following benefits [Ref 17]:

1) Risk reduction by providing a systematic method for validating systems.
2) Reliable code through automation.
3) Reduced maintenance costs.
4) Fewer system integration problems by standardizing all interfaces.

B. THE WARRIOR PROTOTYPE

1. Purpose of the Prototype

We needed to develop an executable prototype to validate our object-oriented Model of the Janus Subsystem. By using CAPS to rapidly create the prototype, we could continue the prototyping process in order to refine the Janus interfaces, adjust the designs to handle newly discovered issues, and exercise all parts of the architecture [Ref. 19].

Thus, the result of the prototype process is to validate the architecture to ensure that the architecture will meet the user's needs.

2. Building the Prototype

When developing an executable prototype of the simulation, we focused on four subsystems: Janus, GUI, JAAWS, and the POST_PROCESSOR. Our architecture did not include re-engineering the JAAWS and POST_PROCESSOR subsystems. Nevertheless, we felt it imperative to include these systems in our prototype to validate the interface among these subsystems. Figure 4 shows the top-level PSDL structure of the prototype.

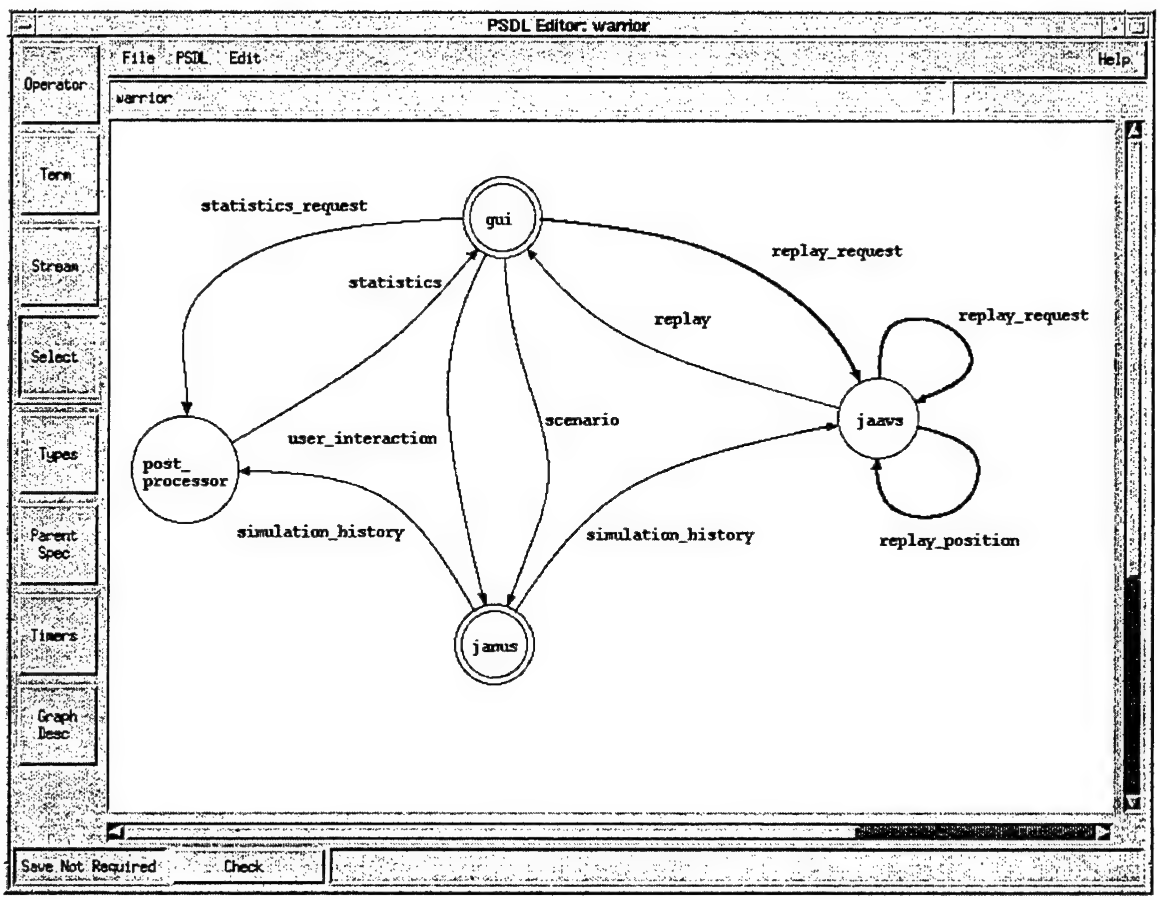


Figure 4. Top-level decomposition of the executable prototype

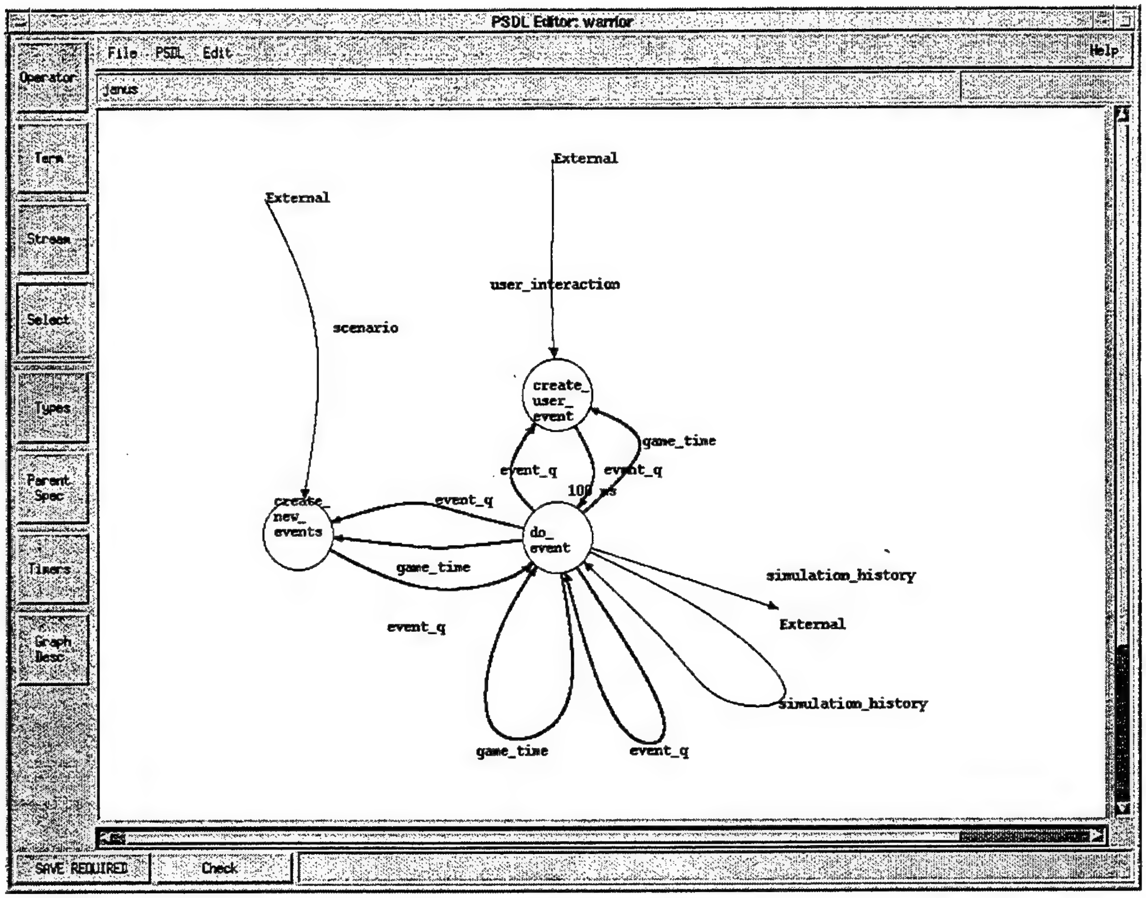


Figure 5. The JANUS subsystem of the executable prototype

Among the four subsystems, the Janus and GUI subsystems, depicted with double circles, are made up of sub-modules as shown in Figures 5 and 6. The POST_PROCESSOR and JAAWS subsystems, depicted with single circles, are mapped directly to objects.

Due to time and resource limitations, we developed the prototype for a very small simulation. In order to fully exercise all parts of the architecture we chose a single object, a Tank, moving on a two-dimensional plane along with three event objects (*Move_Update_Object*, *Do_Plan*, *End_Simulation*), and one Post-Processor related statistic, Fuel Consumption.

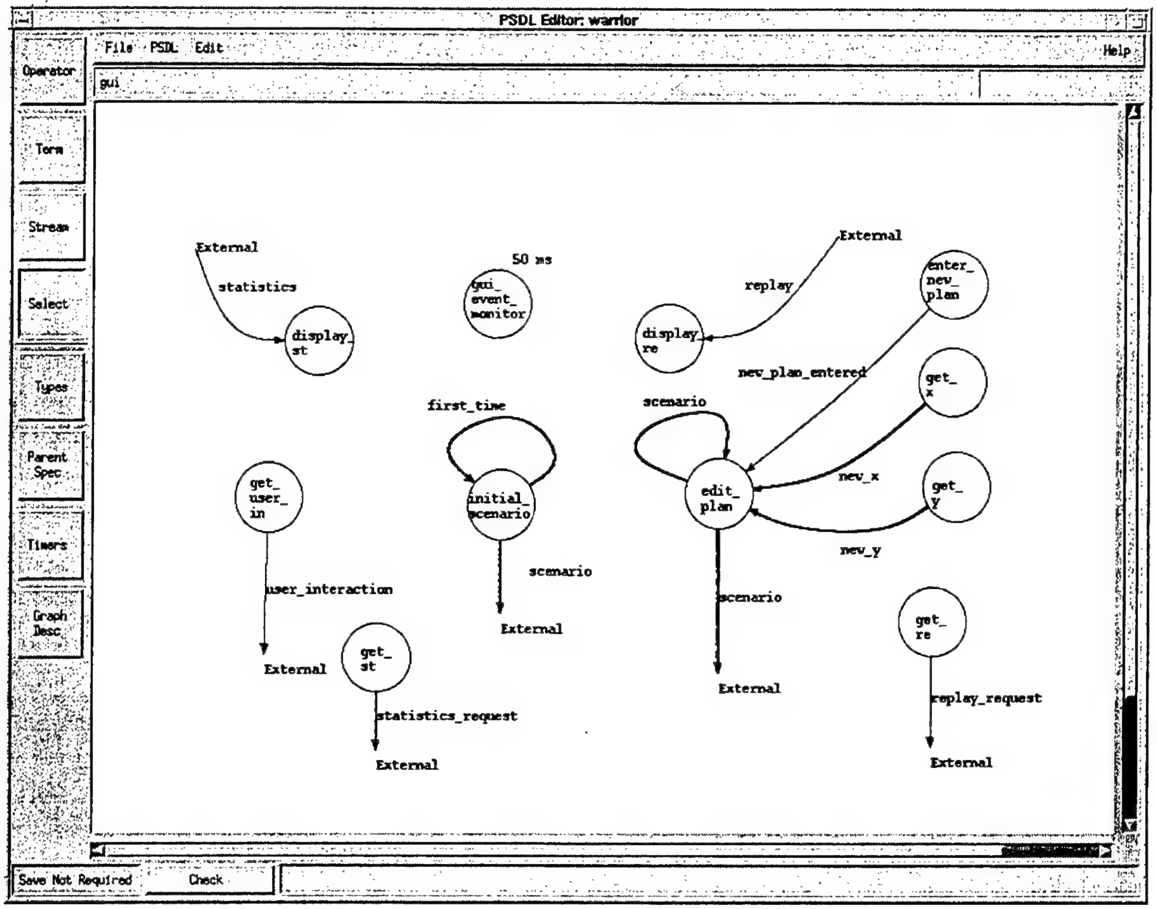


Figure 6. The GUI subsystem of the executable prototype

After creating the PSDL specification of our prototype design (Appendix F), the CAPS execution support system was able to generate the code that controls and interconnects the subsystems. By providing functionality to the subsystems, we were able to generate an executable prototype. When adding functionality, we were careful to ensure we conformed to the object model we developed in a prior exercise. For example, our design of the event handler required the handler to be able to execute events for all kinds of simulation objects. In our prototype, this meant that the *Move_Update_Object* event handler had to work with all different kinds of simulation objects. Although our tank only moved in two dimensions, we choose to implement the *Move_Update* method

of the base Vehicle Class to support 3-D movement because some objects must have the capability to move in three dimensions. Using the object-oriented property of polymorphism, we allowed the event handler to invoke the correct implementation based on the object type, thus solving this problem. In addition, using the Transportable Application Environment (TAE)[2], we were able to develop a simple GUI to allow easy access and execution of the prototype. The resultant prototype consisted of over 6000 lines of program source code and contained enough features to exercise our architecture (Figure 7 and Appendix G).

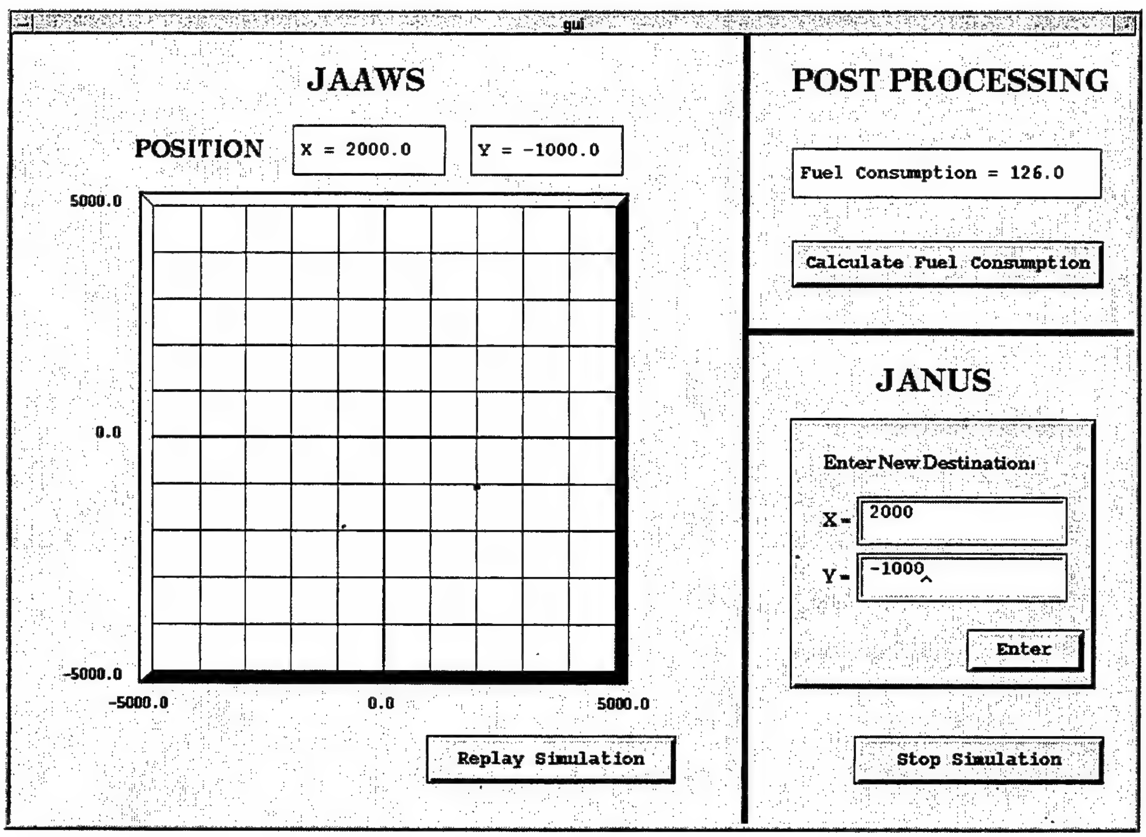


Figure 7. The Graphical User Interface of the executable prototype

[2] TAE is a trademark of the National Aeronautical and Space Administration.

C. PROTOTYPE REFINEMENTS

As we worked through the prototyping process we made several refinements to our architecture. The prototype resulted in the following refinements:

1. Return Value of the Execute_Event Method

Our first refinement was to change the *Execute_Event* method's return value from a null value to a value representing the time to reschedule the next event for the simulation object. In the original implementation, the *Do_Event* method retrieved an event object from the queue, which was then dispatched to the correct *Execute_Event* method via polymorphism. The *Execute_Event* method then executed the event on the corresponding simulation object. If, while in the *Execute_Event* method, the object required rescheduling, it made an external call back to the *Do_Event* package to invoke the *Schedule_Event* method, thus adding the event object to the queue.

However, we felt that all queue operations should be localized to the *Do_Event* package and that execution of events should be localized to the *Execute_Event* package. Therefore, by implementing the return value, we were able to separate these operations (Appendix G). Now instead of the *Execute_Event* method making an external call back to the *Do_Event* package to reschedule the event, the method just returns the time to reschedule that event if necessary. Once control is returned to the *Do_Event* method, it will reschedule the event. We introduced a special time value of "*NEVER*" to indicate that the event should not be rescheduled. The proposed modifications changed communications between the event dispatcher and the simulation objects from a peer-to-peer type relation into that of a client/server type relationship. This change also served to reinforce the object-oriented property of information hiding by eliminating the need for

the simulation objects to know the details of the event queue. It additionally reinforced the property of polymorphism by allowing the dispatcher to use a single statement to schedule all recurring events for all event types including those that may be added by currently unknown future extensions to Janus.

2. Simulation History

Our second refinement to the architecture dealt with how to record the history of the simulation run. Instead of recording the history in terms of periodic snapshots of selected data values, we decided to record the simulation history as a sequence of events. Our first implementation of recording the history was based on the current Janus model. In this model, a special event, *Write_Status*, executed periodically to capture selected data values and write them to disk. However, the prototype showed us that this model was highly inefficient, inaccurate and often unreliable. For example, many of the simulation objects' states remained constant through several *Write_Status* executions. Most of the data captured was redundant. In order to increase the accuracy, the system would have to capture more data values. Combined with the redundant data problem, one could easily see how the size of the history file could easily grow rapidly. Moreover, the *Write_Status* event handler has to keep track of the status of the objects that own the selected data values.

After further investigation, we realized that the state of the simulation only changed when an event occurred. If we were able to capture each event in a history file, we could then capture a true representation of the simulation as it occurred. Since we were already using the event objects for the real-time simulation, using the event objects to create a historical record provided a simple and uniform way to conduct post-

simulation analysis. As we implemented this change, we discovered two primary benefits. First, we were able to provide the post-simulation with the greatest resolution without creating an excessively large database. By capturing the events, any quantity calculated during the real-time simulation could also be calculated during the post-simulation. Second, we eliminated the need for a *Write_Status* event from our architecture. Instead of using the *Write_Status* event (Appendix B) to capture the history, we made a single line modification to our *Do_Event* method (Appendix G, Section 25). After an event is executed, a copy of the event is placed in the *Simulation_History*.

3. Null Action of an Event in the Event Queue

Our third refinement allowed the null action of an event to appear in the event queue. A null action of an event does not affect the state of the simulation but serves as a placeholder of a dormant object in the event queue and also serves as a method to allow future events. In our first version of the prototype, we opted to not allow null events into the event queue since this decision corresponds to the current Janus scheduling policies. As implemented, we used the *Create_New_Events* method to scan through all of the simulation objects once per simulation cycle to determine if any dormant objects became active. If so, after determining the object and correct corresponding event, the *Create_New_Events* method would reschedule the object in the queue.

The prototype revealed that this process required very complicated logic and greatly reduced the efficiency of the system. By allowing a null action of an event in the event queue, we eliminated the need for the *Create_New_Events* method to scan through all of the simulation objects. We now put the responsibility on the event handlers to manage dormant objects. Constructors for all kinds of simulation objects are used to

create the initial events in the queue. As described above, the *Execute_Event* method of each event handler determines the next time to execute the event and returns it to the *Do_Event* method for rescheduling. However, if the *Execute_Event* method of an event handler determines that the object is not yet active, it simply returns an estimated time to the *Do_Event* method. The *Do_Event* method will then invoke the event of the object again at some time later in the simulation. Then, in the future when the event waiting in the queue executes and the object becomes active, the event handler would determine the correct action and return a schedule time for the meaningful event. For instance, if a vehicle arrives at its destination, the *Move_Update_Object* event handler would flag its event to do nothing and wait for some duration of time to allow the user to provide a new destination. If this duration were not long enough, it would repeat this waiting process. As soon as a user provided a new destination, the *Move_Update_Object* event handler would return a value and enable its event to move the vehicle.

In its final version, the prototype showed this refinement greatly improved overall system efficiency and simplified the code in the following areas:

1) Checking to verify if a dormant object became active is done once per activity of that object instead of once per simulation cycle.
2) Null actions of events are fast since they are basically just a guard. The time required to check a guard in a Null action of an event once per activity is much less than that required to check the status of each simulation object once during each simulation cycle.
3) Eliminating complicated code in order to find and test all simulation objects reduces the computational load on the system.

D. LESSONS LEARNED

Throughout our prototyping experiment, we learned the benefits of using the prototype process. Each iteration of the prototyping process produced several criticisms, many of which resulted in the refinements described above, which ultimately produced a more realistic prototype. In the end, we developed a prototype that successfully met the users needs and validated our architecture.

We observed the many benefits of designing a prototype using an object-oriented architecture. In the course of two weeks, with the assistance of CAPS, we were able to rapidly build four versions of our prototype. As our prototype evolved, CAPS ensured consistency of each version while the three-tiered object-oriented architecture allowed localization of design issues and provided an easy means for extensions. For instance, version 1 of our prototype consisted of only two event subclasses, *Move_Update_Object* and *End_Simulation*. Version 2 introduced a third event subclass, *Do_Plan*. The *Do_Plan* event allowed the user to select new destinations for the tank. Because of the unique flexibility of our object design, we were able to add this event without having to modify the event control loop, thus unaffecting the previously working code. After we implemented *Do_Plan*, we discovered that our implementation forced the user to first enter an X-coordinate followed by a corresponding Y-coordinate. Since our object-oriented design localized the implementation of the *Do_Plan* event to just a few lines of code, we were able to find, fix, and implement a better *Do_Plan* event. In the final state, we added an Enter Button to the GUI to allow the user to enter the coordinates in any order or to allow the user to change one coordinate at a time.

In the final analysis it is easy to see that our architectural design benefited immensely by our decision to use an object-oriented methodology and design. The use of multi-tiered architectural concepts within our three-tier architecture allowed for isolation of the Applications layer into separate components. This is very valuable in the sense that it promotes the use of reusable components and allows for the distribution of tiers on different physical computing hosts. Another benefit is that it provides flexibility to allow different developers to construct specific tiers of the architecture, as is the case in our project where a separate contractor is developing the system's GUI.

Additionally, the object-oriented properties of polymorphism and inheritance greatly enhanced our ability to efficiently extend the behaviors and provide specific behaviors to objects. For example, in version 2 of our prototype, we introduced the POST_PROCESSOR module, which allowed users to view the vehicle's fuel consumption. However, a fuel consumption calculation would require more specific behavior from our tank's *Move_Update_Object* method. In addition, as described above, in version 1 of our prototype we used polymorphism and inheritance properties to move the tank, thus allowing the event handler to dispatch to the Tank Class. This again demonstrated the unique object-oriented luxury of inheriting general-purpose movement behaviors from the superclass. In version 2, we had to modify this behavior to show the vehicle's fuel consumption. By calculating the time elapsed from the last movement and using the vehicle's fuel consumption rate per time, we were able to compute the correct amount of fuel consumed. In the end, the Tank Class *Move_Update_Object* method consisted of a guard, one statement to invoke the superclass *Move_Update_Object*, and three lines to calculate the amount of fuel consumed (Appendix G, Section 81). Again

using the properties of polymorphism and inheritance, additional objects such as helicopters, airplanes and other mobile weapon platforms can be easily implemented in a similar fashion.

V. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

Our thesis research indicates that by applying reverse engineering, restructuring, and forward engineering techniques, we were able to successfully evolve a procedurally coded, function oriented legacy software system into a highly modular, object-oriented software system capable of contending with current simulation systems. We developed an architecture that supports the functionality of the current Janus simulation system, while also maintaining the flexibility to evolve to handle new and future design changes. Using object-oriented design and analysis techniques, we created objects and classes that encapsulated related items and developed a structural model of the system. This enabled future modifications of the system to be accomplished by manipulating those objects or by introducing new objects into the system. For example, while working on the Janus simulation system, TRAC-WSMR informed us that the Janus model for the Radar implementation was outdated and that a new model would be out within a year. However, we had already completed this section of our architecture and based it on the currently outdated Radar model. Nevertheless, since we used an object-oriented architecture, it will be relatively easy to incorporate any changes to our model since all of the information regarding the Radar functionality is located in only two areas, the Red and Blue Radar objects.

B. RECOMMENDATIONS

1. Automated Tools

As we look back on the Janus simulation re-engineering process, we can recall the enormous amount of time and effort spent analyzing the system. This process was a literal nightmare consisting of manually intensive sessions reading and tracing through FORTRAN source code, reading technical and user manuals, not to include several sessions spent actually running the Janus simulation software. Although this was an extremely important phase of our project and one that could not be overlooked, it was nevertheless, very time consuming and strenuous. The use of some well-defined automated tools would have been helpful in gaining system understanding. Although one can not assume that any one tool will be a "be all, do all", but rather having even a small collection of tools will assist the software engineer in gaining system comprehension. As such, we highly recommend the use of automated tools to aid in reducing the time and effort spent analyzing the system, and also to assist the software engineer in examining the raw data.

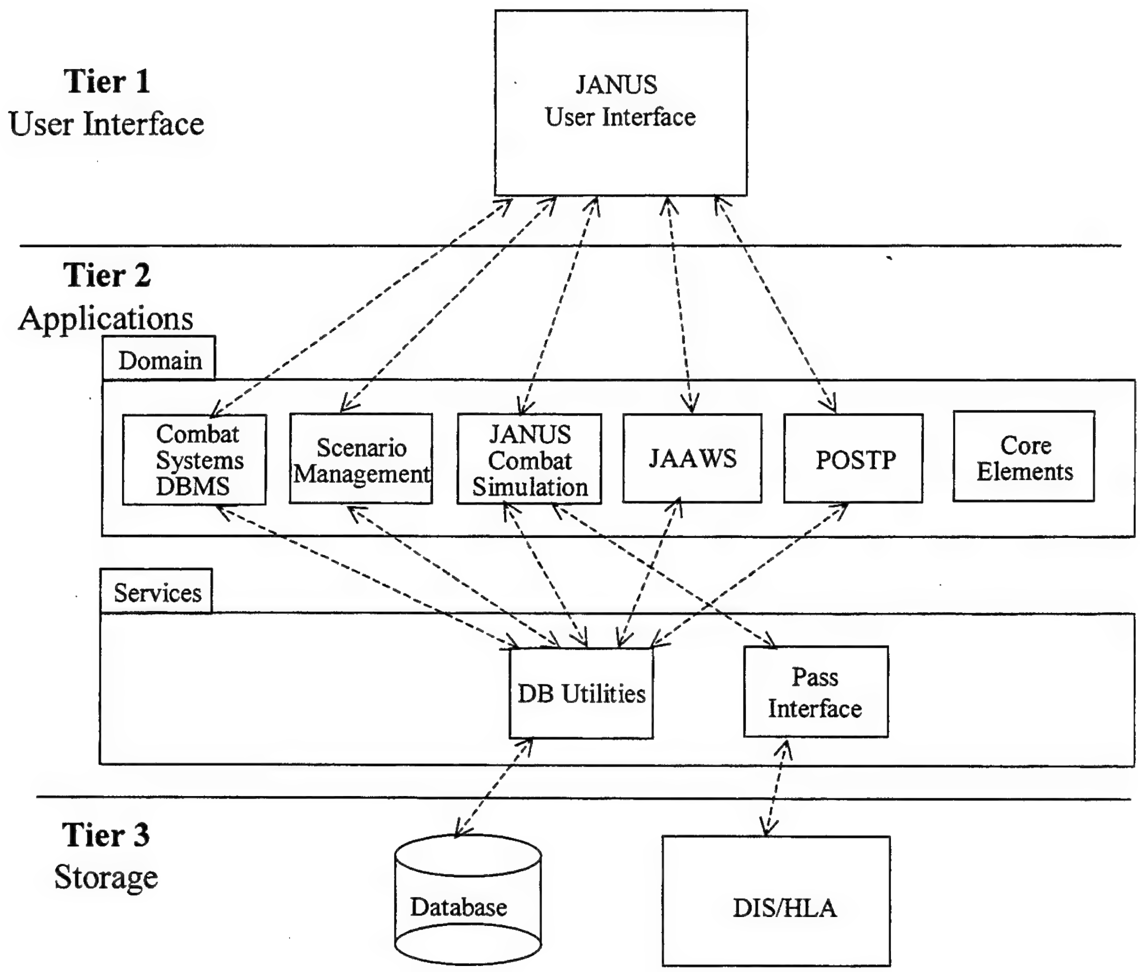
2. Cross-Reference Generators

As part of our research we often found ourselves manually searching through the Janus source code to find answers to specific questions regarding a particular variable (i.e. who uses it, where is it used, how is it used, etc.). Professor Berzins suggested using the cross-reference option, which was readily available on the CS Department's FORTRAN compiler. A cross-reference generator creates a list of all of the identifiers in a program and for each identifier in the program, it indicates the statement in which that identifier appears. The use of a FORTRAN cross-reference generator would have been very helpful in gathering needed variable information and thus assisted us in understanding the system. Even though we could not compile and run the Janus simulation system on our SUN UNIX machines, the FORTRAN compiler could have automatically built a cross-reference listing of all variables. Then by using this cross-reference, we could quickly and easily find answers to our specific questions and thus eliminate wasteful, time-consuming manual searches.

Throughout the conduct of our thesis research, few areas stand out as those requiring a high degree of focus and intense attention to detail. System understanding was one such area that demanded intensive manual activity and time. Techniques to automate this process would allow the developers to devote more time and energy to the overall design and implementation of the new system.

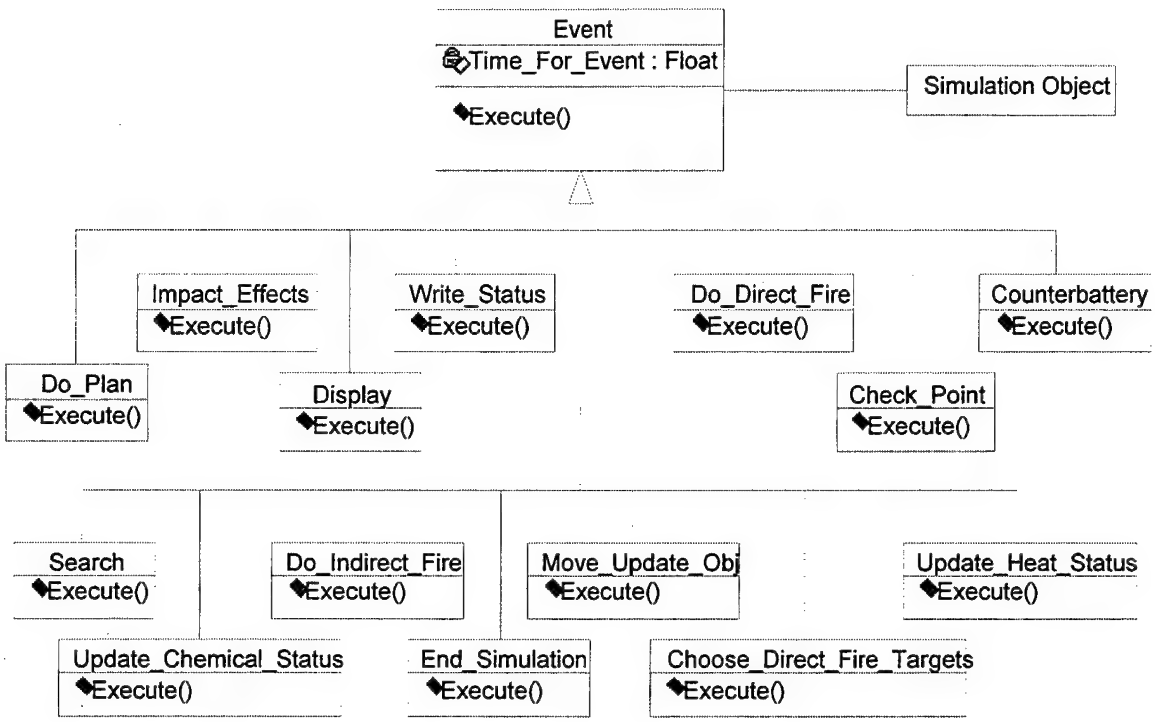
THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. PROPOSED THREE-TIER OBJECT-ORIENTED ARCHITECTURE



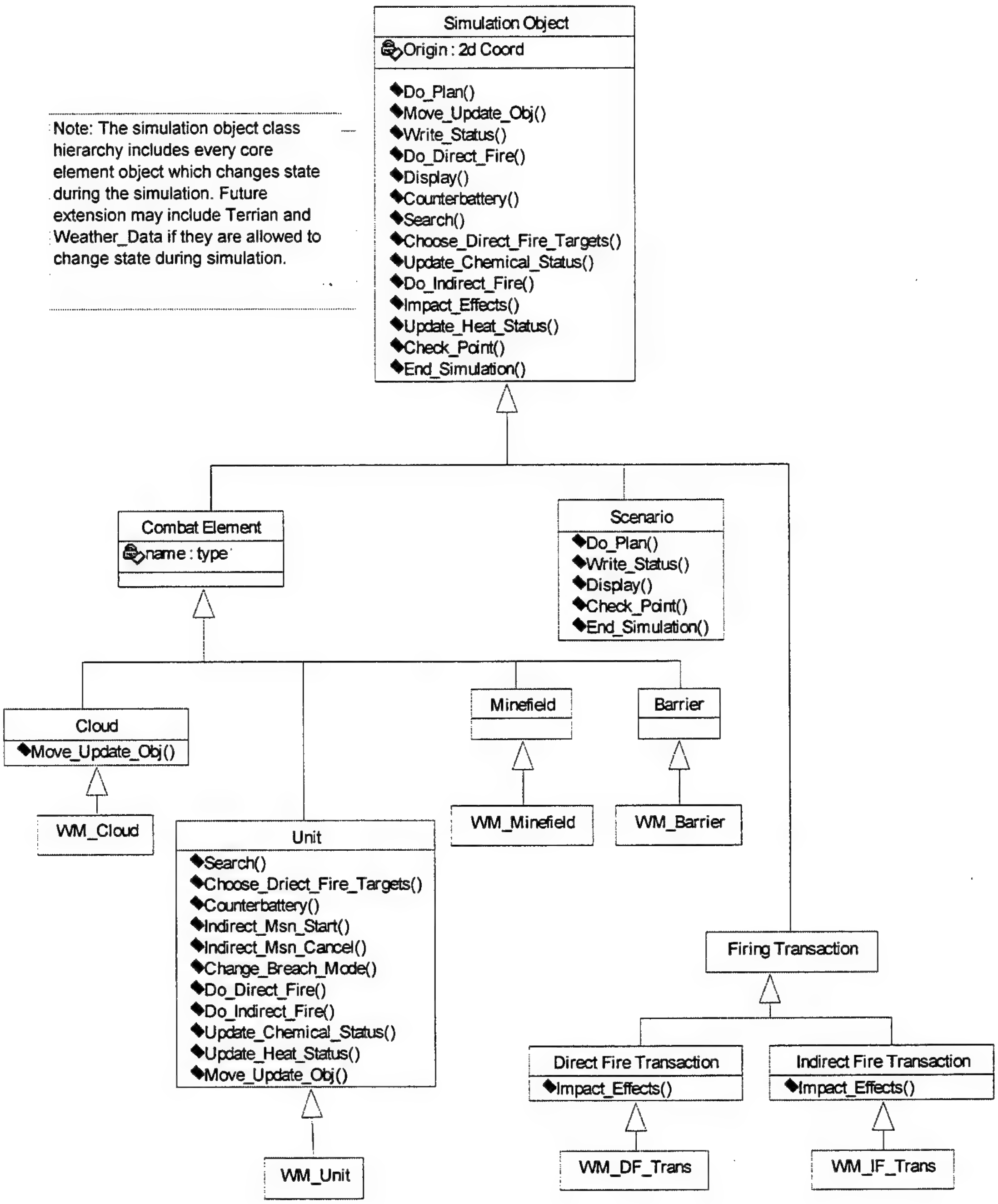
THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. EVENT CLASS HIERARCHY



THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. SIMULATION OBJECT CLASS HIERARCHY



THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. JANUS SIMULATION EVENT HANDLERS

Name of Event	Objects responsible to handle the event	Remarks
Do_plan	scenario	May be initiated by the graphical user interface and it in turn may schedule other events. See Note 1.
Display	scenario	May be triggered at regular time interval. See Note 1.
Write_Status	scenario	May be triggered at regular time interval. See Note 1.
Check_Point	scenario	May be triggered at regular time interval. See Note 1.
Move_Update_Obj	Unit	Updates units position due to movement, and schedule the next Move_Update_Obj event for the object. Mapped to movement.f, copter.f, update.f updsldr.f, updflyer.f
	Cloud	Updates shape, location (if needed), and expiration time. It schedules the next Move_Update_Obj event for the object. Mapped to part of docloud.f, cldupd.f, and chmcld.f.
Search	Unit	Update potential target list. Use different methods depending on the kind of sensors the unit has. It also schedules the next search event for the object. Mapped to search.f, part of radar.f for normal and special radar.

Choose_Direct_Fire_Targets	Unit	Updates visibility levels and performs IFF to produce confirmed target list. Selects weapons for the targets in the potential target list. Choose target from the confirmed target list and schedule a direct fire event. Use different methods depending on the kind of platform the unit belongs to and the kind of sensors the unit has. It also schedules next Choose_Direct_Fire_Targets event for the object. Mapped to dodetect.f, detect.f, flydetc.f, handoff.f, and reload, and part of radar.f (for normal radar).
Do_Direct_Fire	Unit	Creates a Direct_Firing_Transaction object and schedules an Impact_Effects event for the object. Mapped to shoot.f , and adfire.f for normal radar.
Do_Indirect_Fire	Unit	Execute an arty mission. Creates an Indirect_Firing_Transaction and schedule an Impact_Effects event for the object. Mapped to doarty.f, and part of firing.f. Event scheduled by Do_Plan event.
Impact_Effects	direct_firing_ transaction	Evaluate the effect of the direct fire event and update the affected objects accordingly. Mapped to dfmpact.f.
	indirect_firing _transaction	Evaluate the effect of the indirect fire event and update the affected objects accordingly. May create cloud objects and schedule Move_Update_Obj event for the cloud objects. Mapped to impact.f.
Counterbattery	Unit	Searches for enemy fires and schedules next Counterbattery event for the object. Feedback is provided via Do_Plan event. Mapped to cntrbat.f.

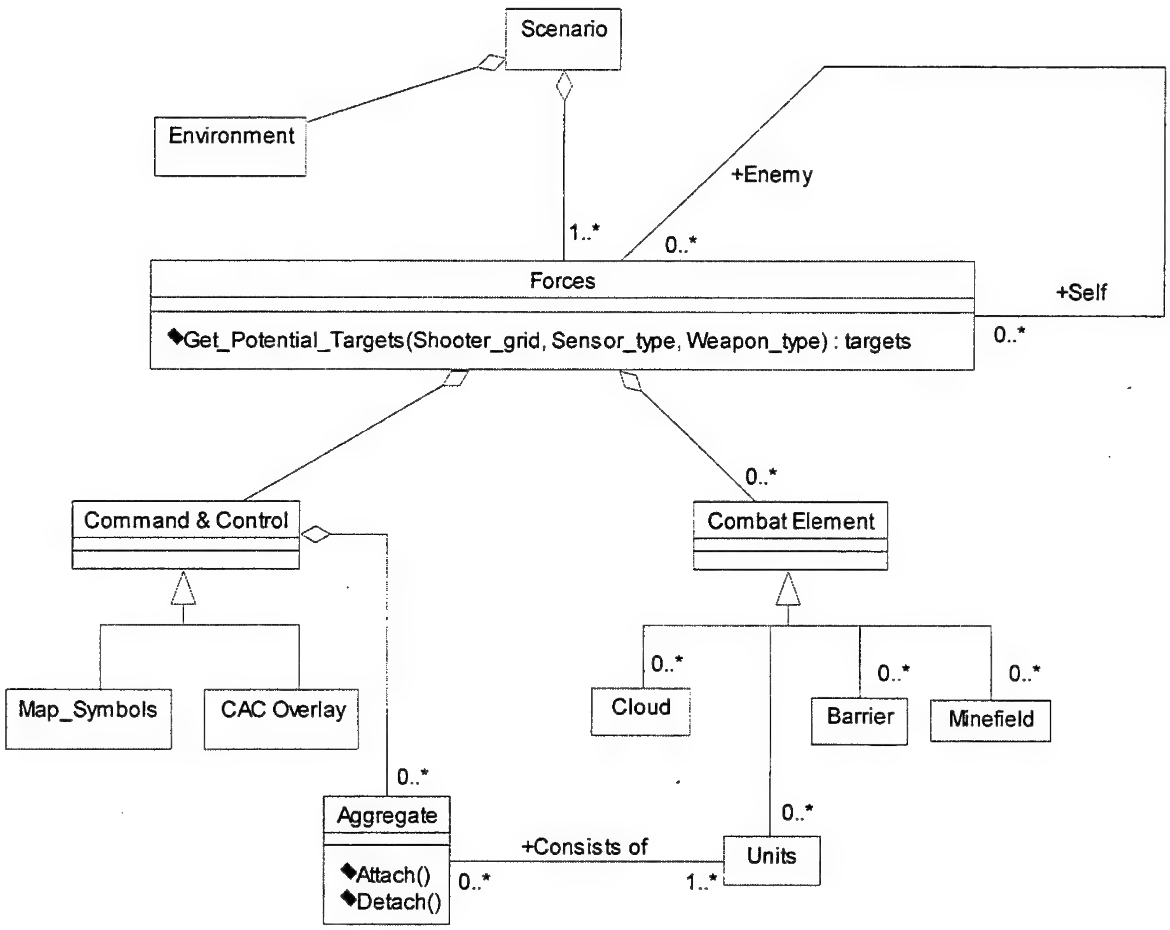
Update_Chemical_Status	unit	Updates chemical effects on unit and schedules next Update_Chemical_Status for the object. Mapped to part of dochem.f
Update_Heat_Status	unit	Updates heat effects on unit and schedules next Update_Heat_Status for the object. Mapped to part of doheat.f
End_Simulation	scenario	Clears the priority event queue and performs housekeeping activities. See Note 1.

Note 1. Depending on the graphical user interface design, this event may be replaced by different events and assign the event handlers to the individual objects.

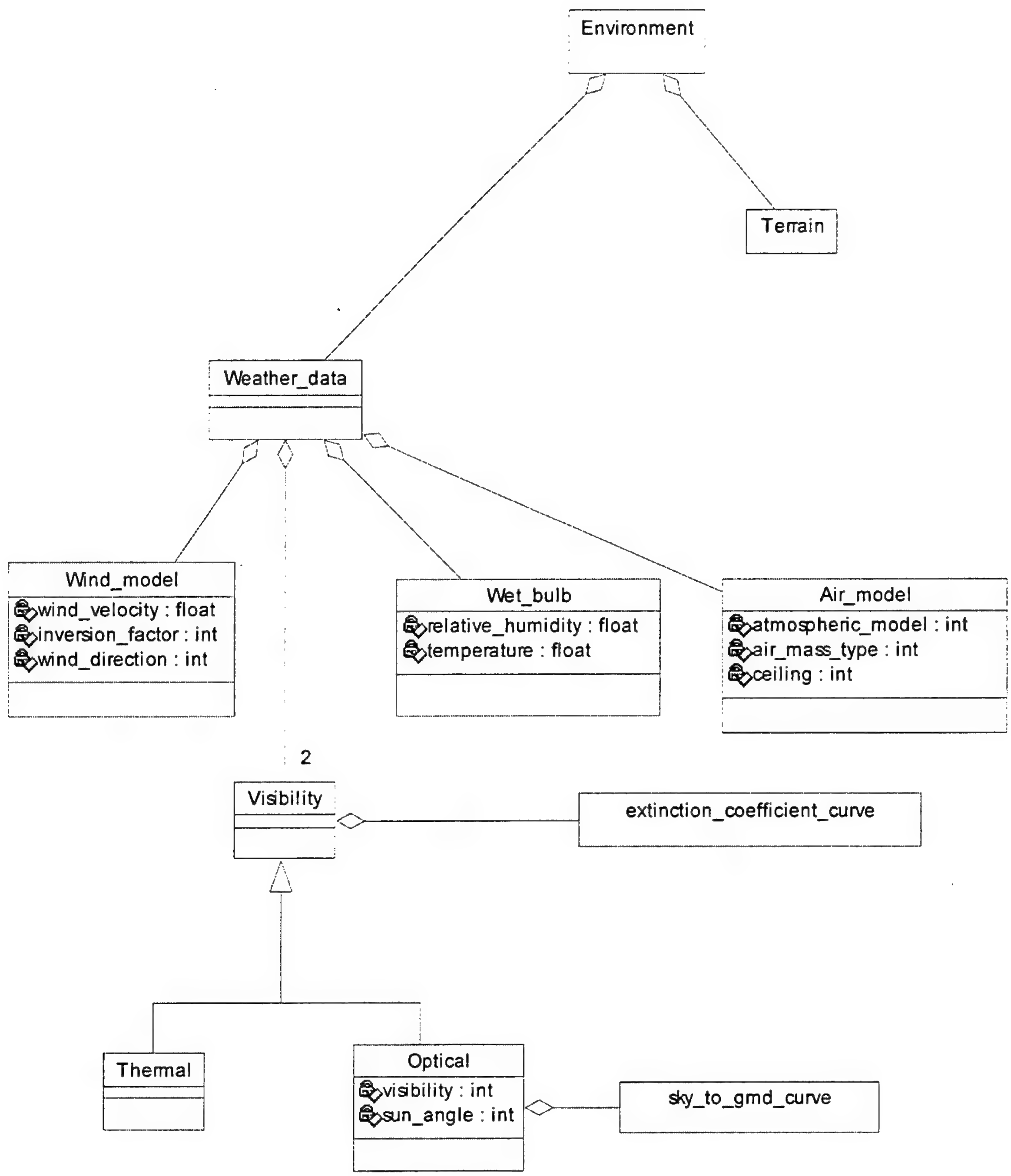
THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX E. JANUS CORE ELEMENTS

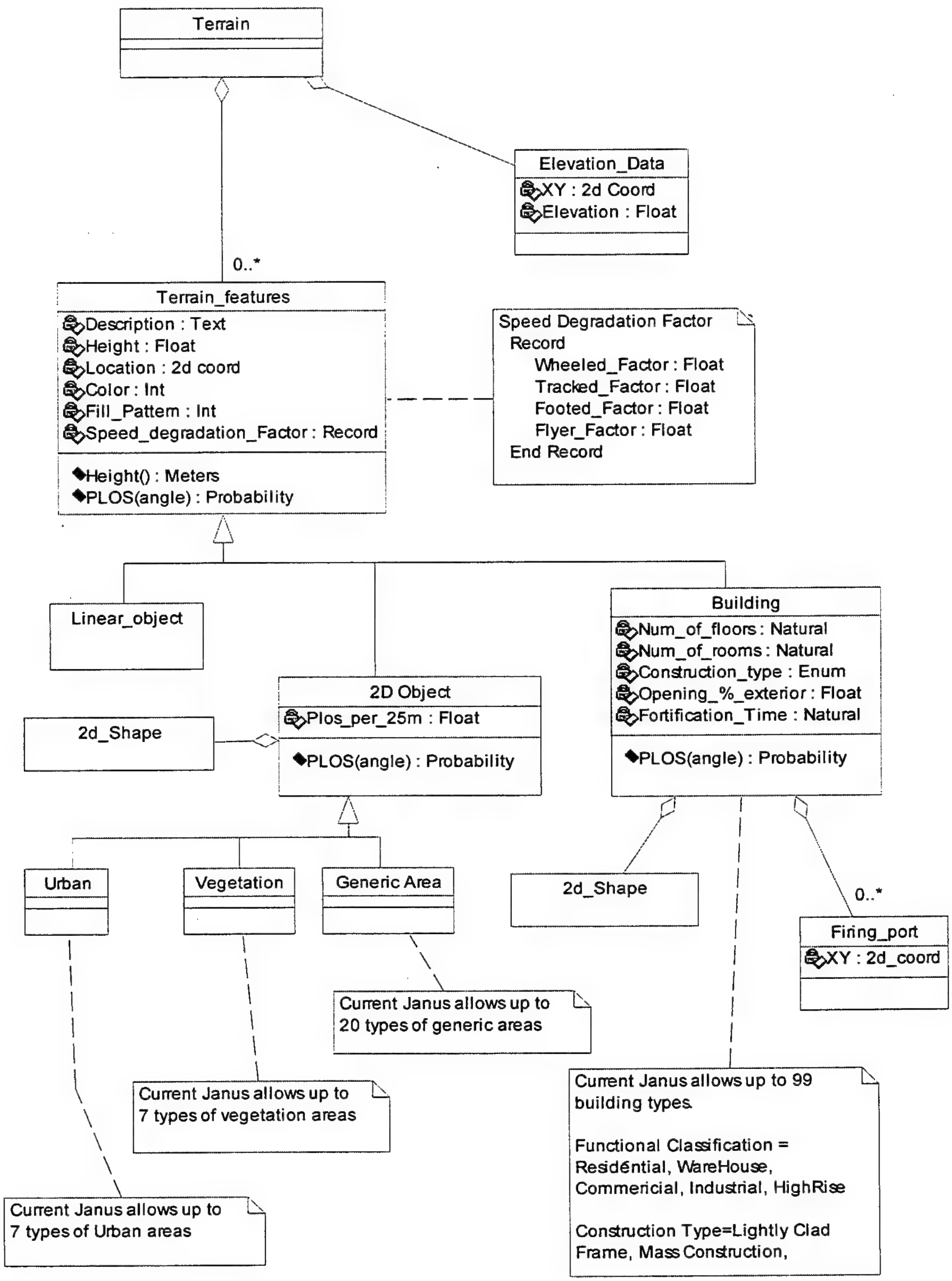
1. JANUS CORE ELEMENTS OVERALL STRUCTURE



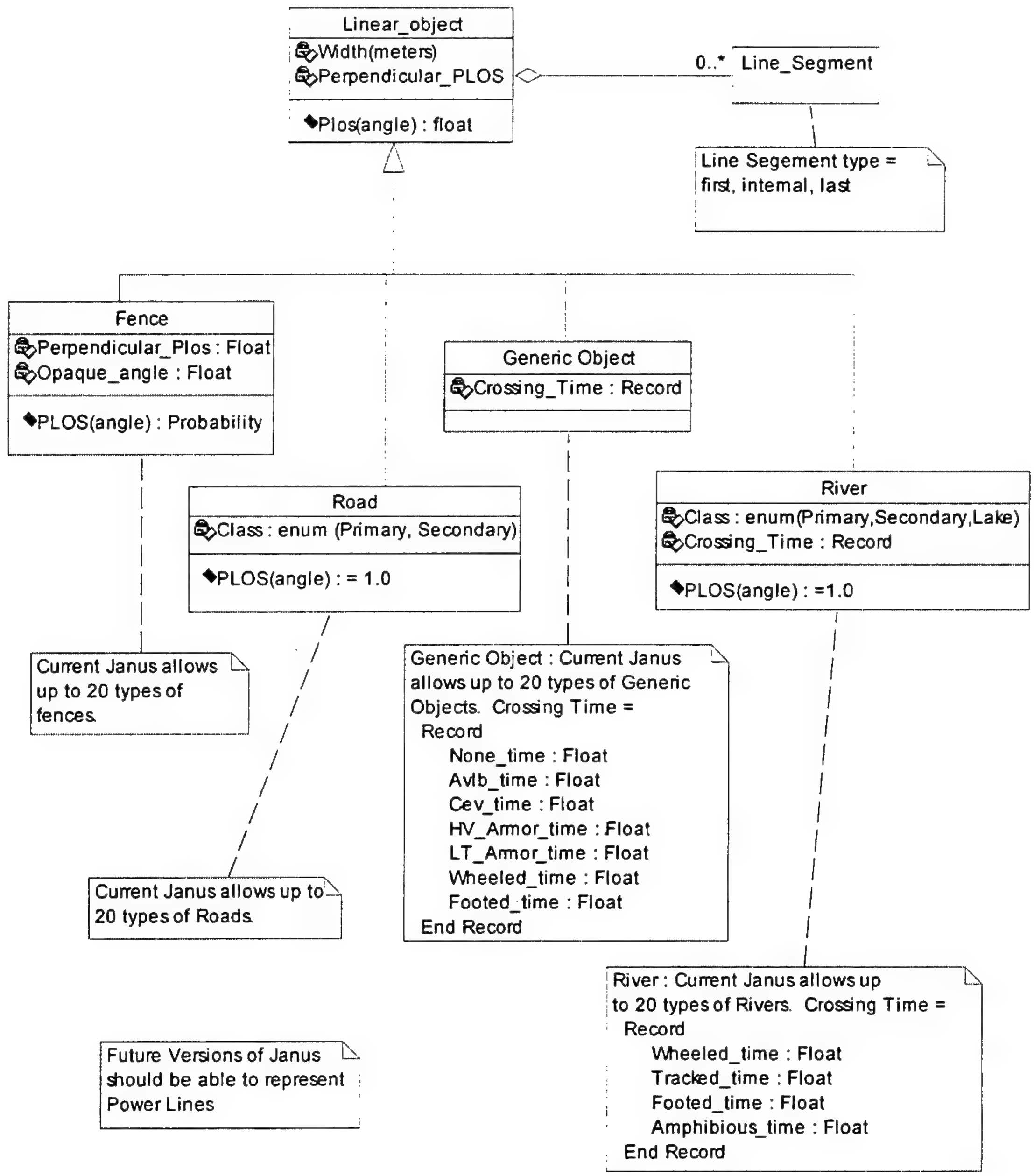
2. ENVIRONMENT AND WEATHER_DATA CLASSES



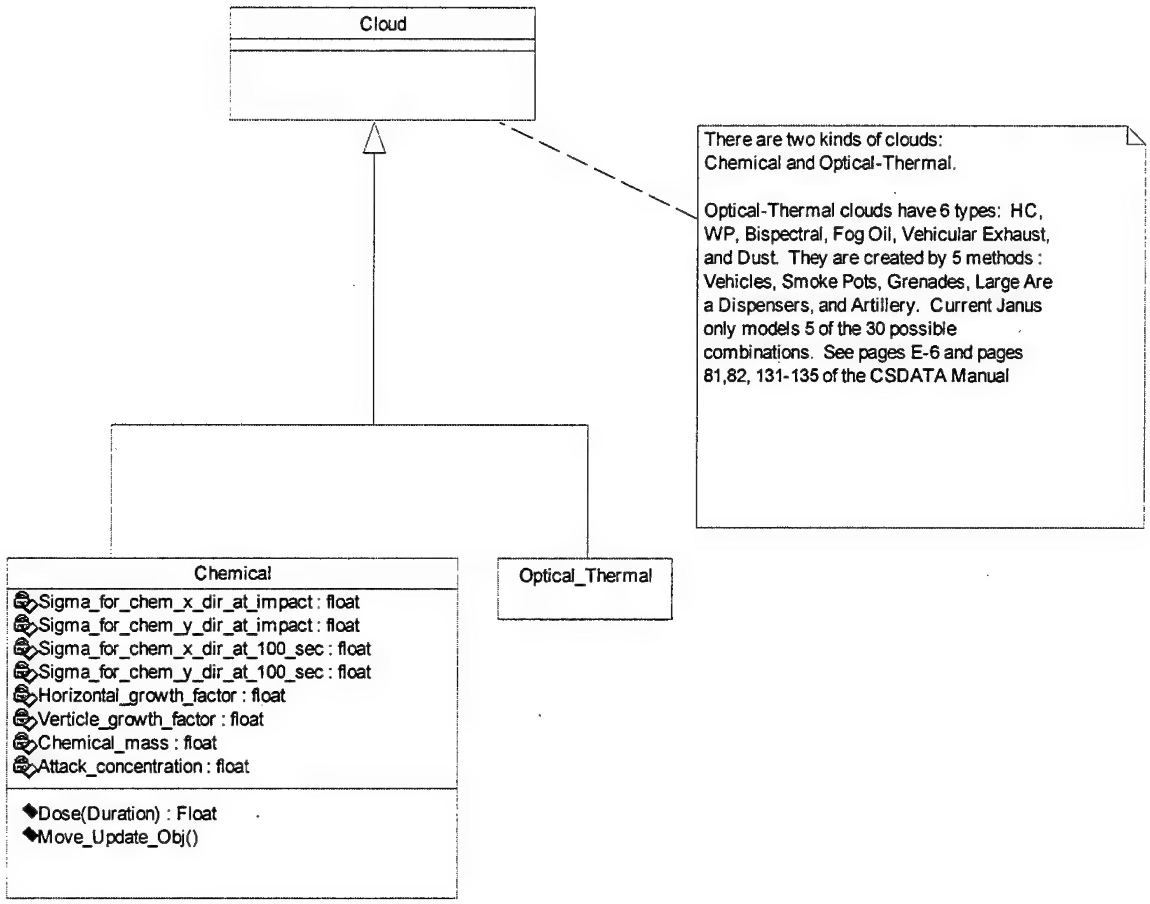
3. TERRAIN CLASS



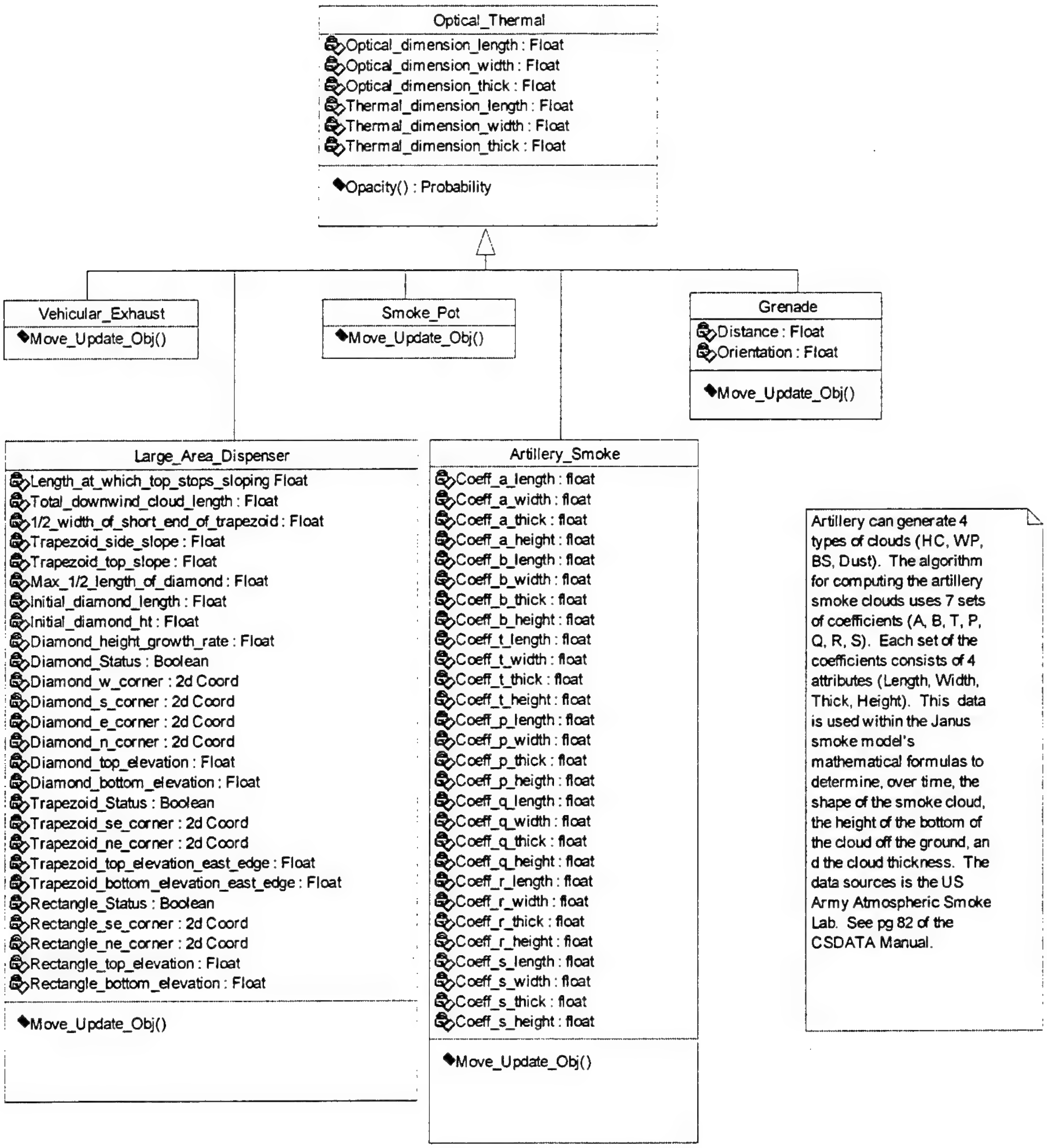
4. LINEAR_OBJECT CLASS



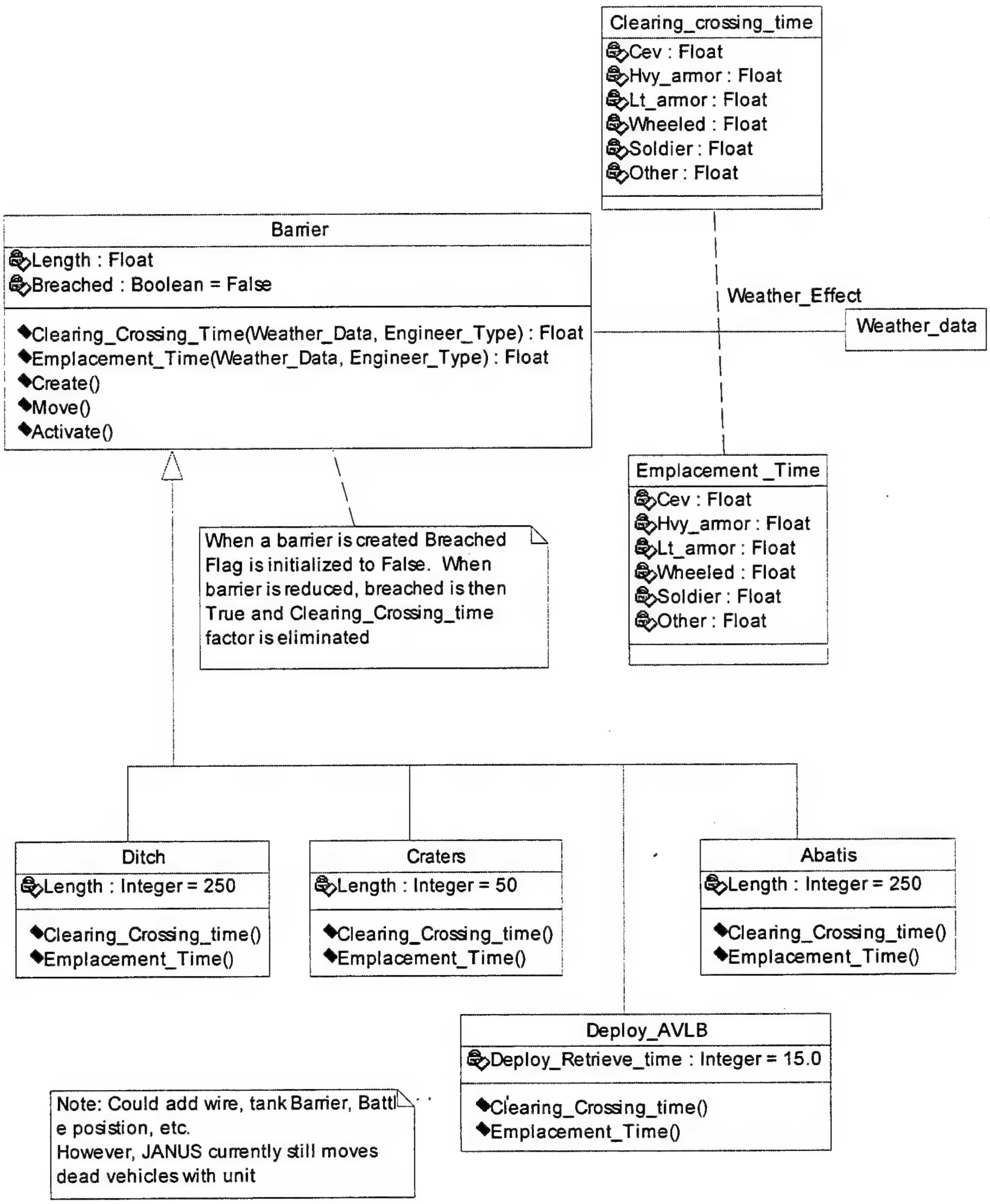
5. CLOUD CLASS



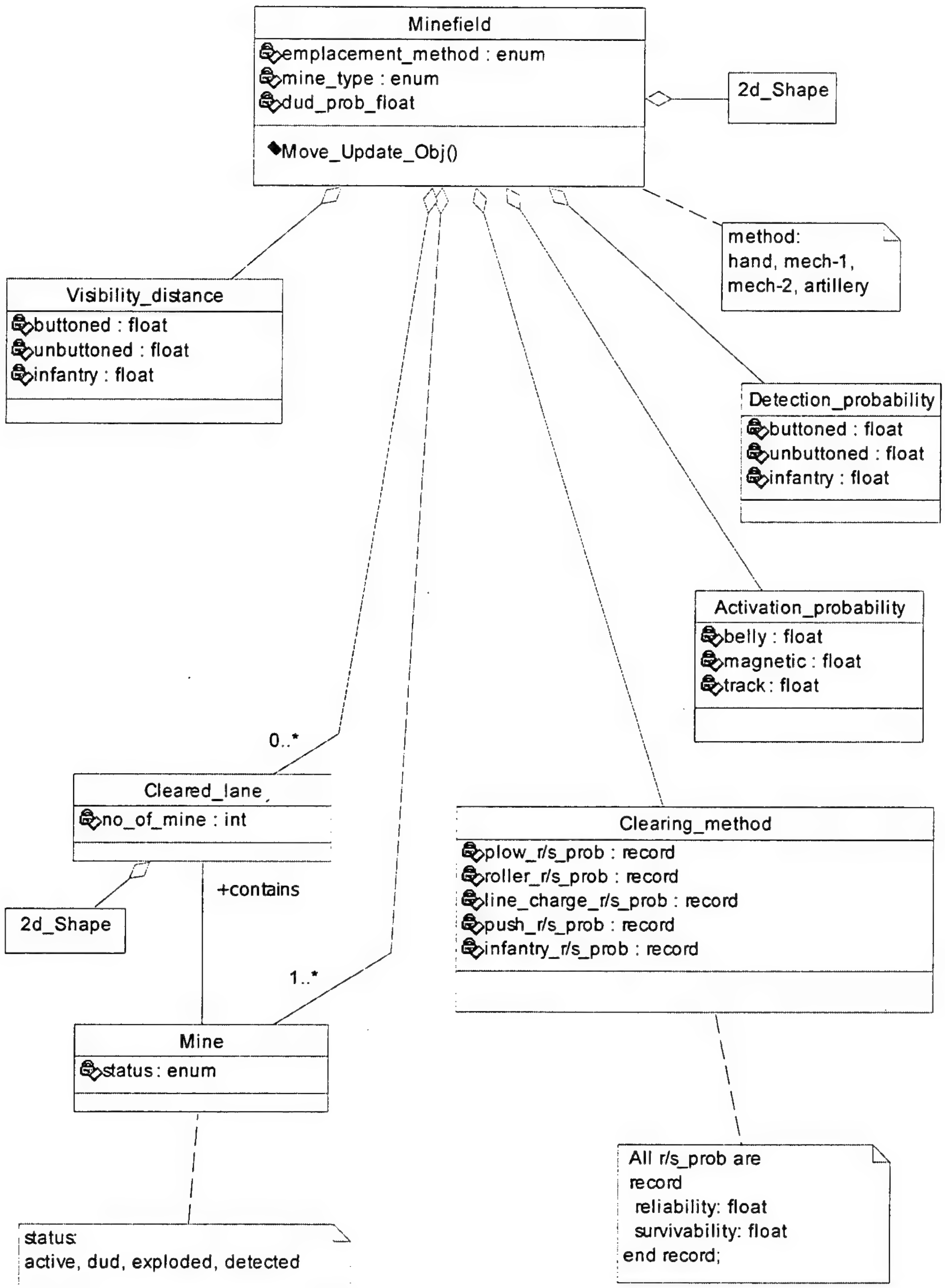
6. OPTICAL_THERMAL CLASS



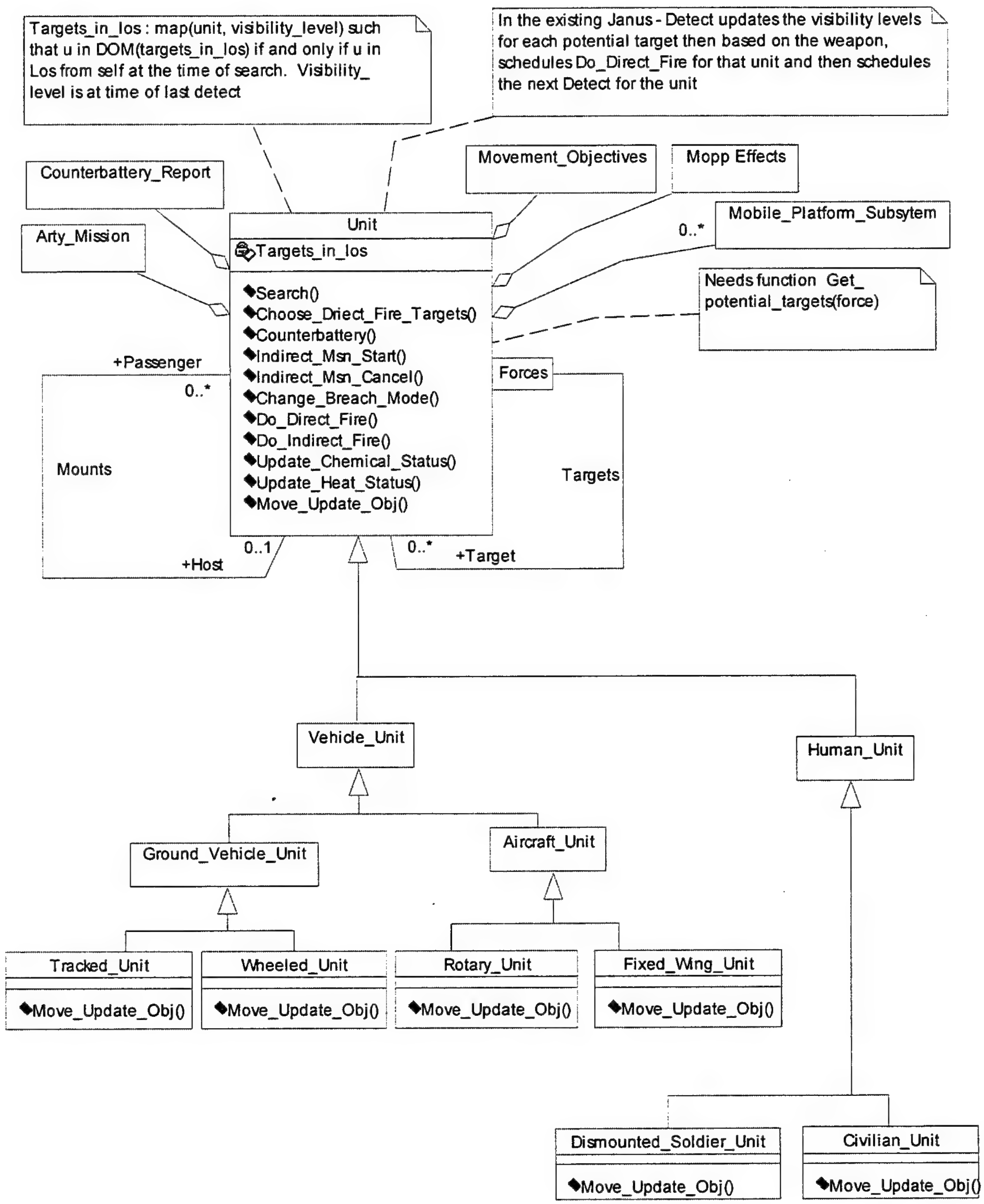
7. BARRIER CLASS



8. MINEFIELD CLASS

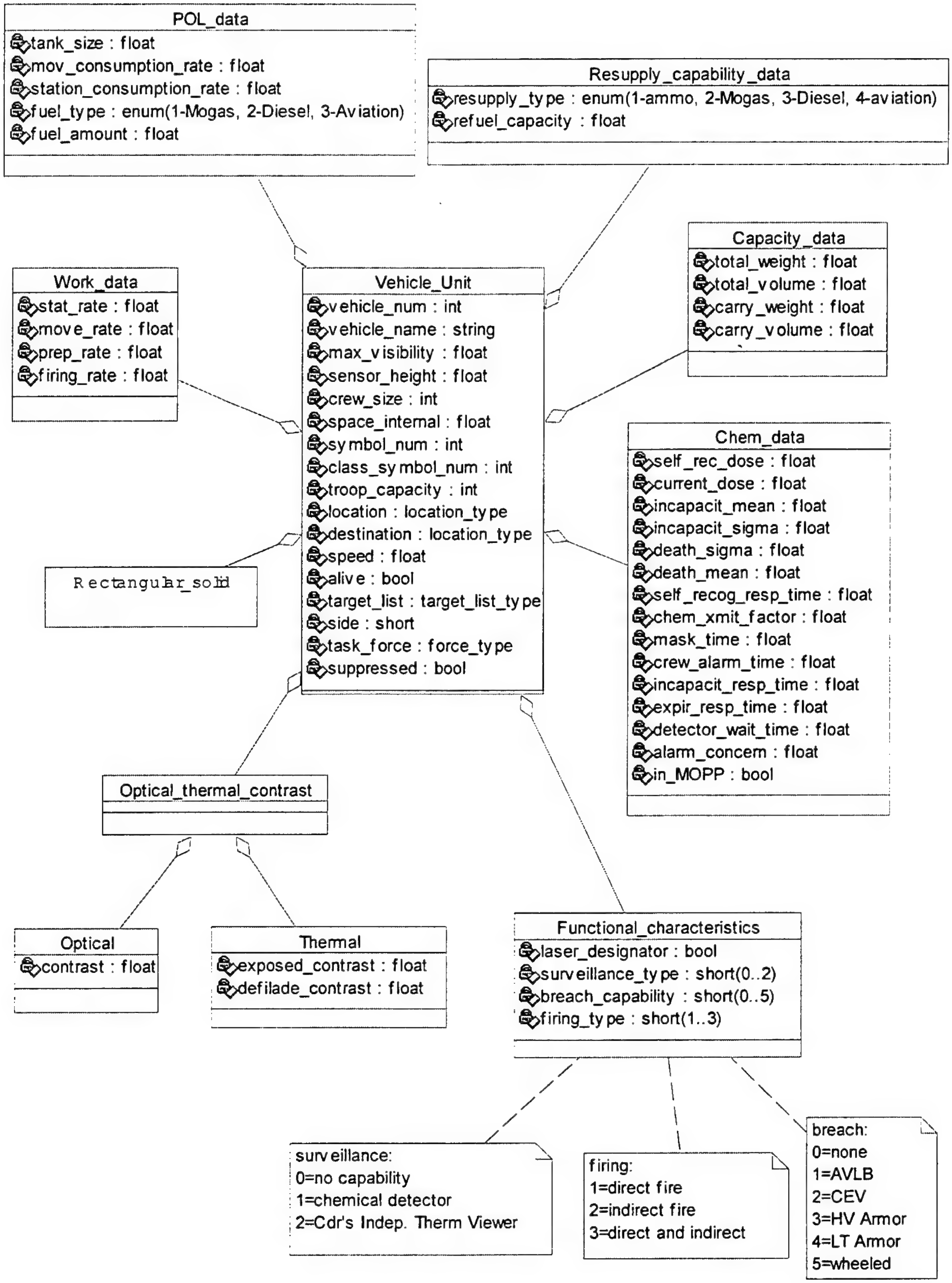


9. UNIT CLASS

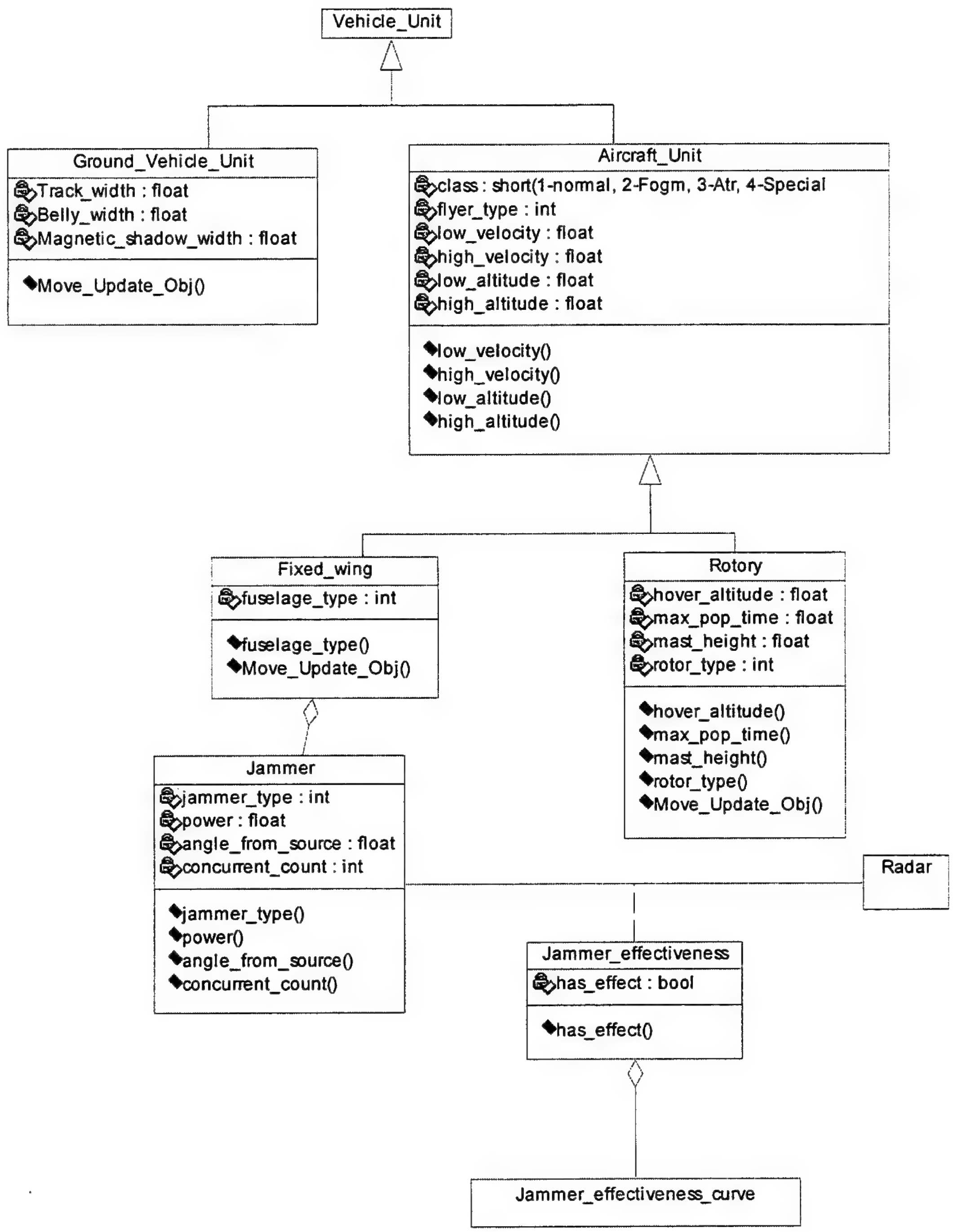


Targets_in_los : map(unit, visibility_level) such that u in DOM(targets_in_los) if and only if u in Los from self at the time of search. Visibility_ level is at time of last detect
In the existing Janus - Detect updates the visibility levels for each potential target then based on the weapon, schedules Do_Direct_Fire for that unit and then schedules the next Detect for the unit
Counterbattery_Report
Movement_Objectives
Mopp Effects
Arty_Mission
Mobile_Platform_Subsytem
0..*
Unit
Targets_in_los
Search()
Choose_Driect_Fire_Targets()
Counterbattery()
Indirect_Msn_Start()
Indirect_Msn_Cancel()
Change_Breach_Mode()
Do_Direct_Fire()
Do_Indirect_Fire()
Update_Chemical_Status()
Update_Heat_Status()
Move_Update_Obj()
Needs function Get_ potential_targets(force)
+Passenger
0..*
Mounts
+Host
0..1
Forces
Targets
0..*
+Target
Vehicle_Unit
Human_Unit
Ground_Vehicle_Unit
Aircraft_Unit
Tracked_Unit
Move_Update_Obj()
Wheeled_Unit
Move_Update_Obj()
Rotary_Unit
Move_Update_Obj()
Fixed_Wing_Unit
Move_Update_Obj()
Dismounted_Soldier_Unit
Move_Update_Obj()
Civilian_Unit
Move_Update_Obj()

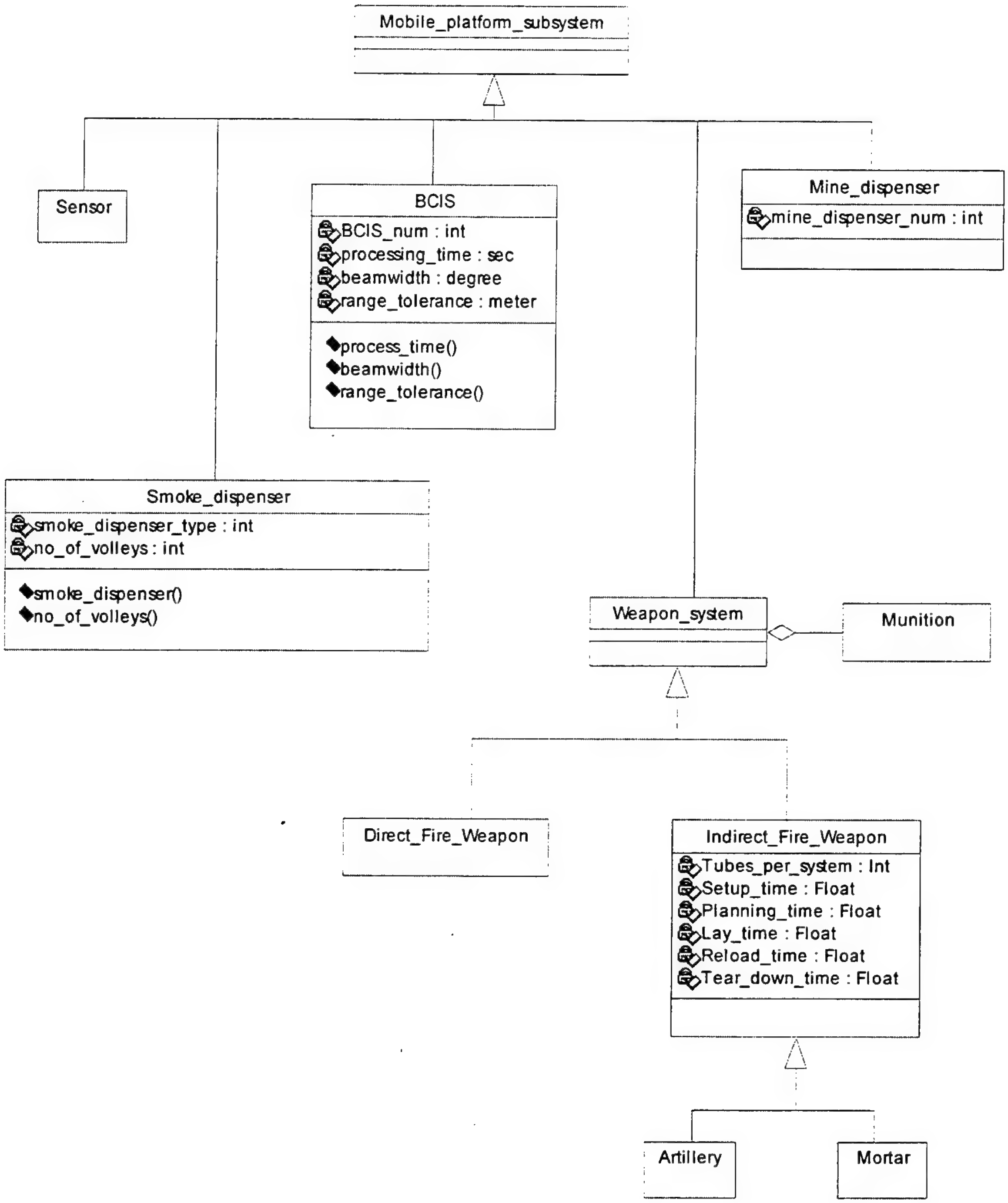
10. VEHICLE_UNIT CLASS



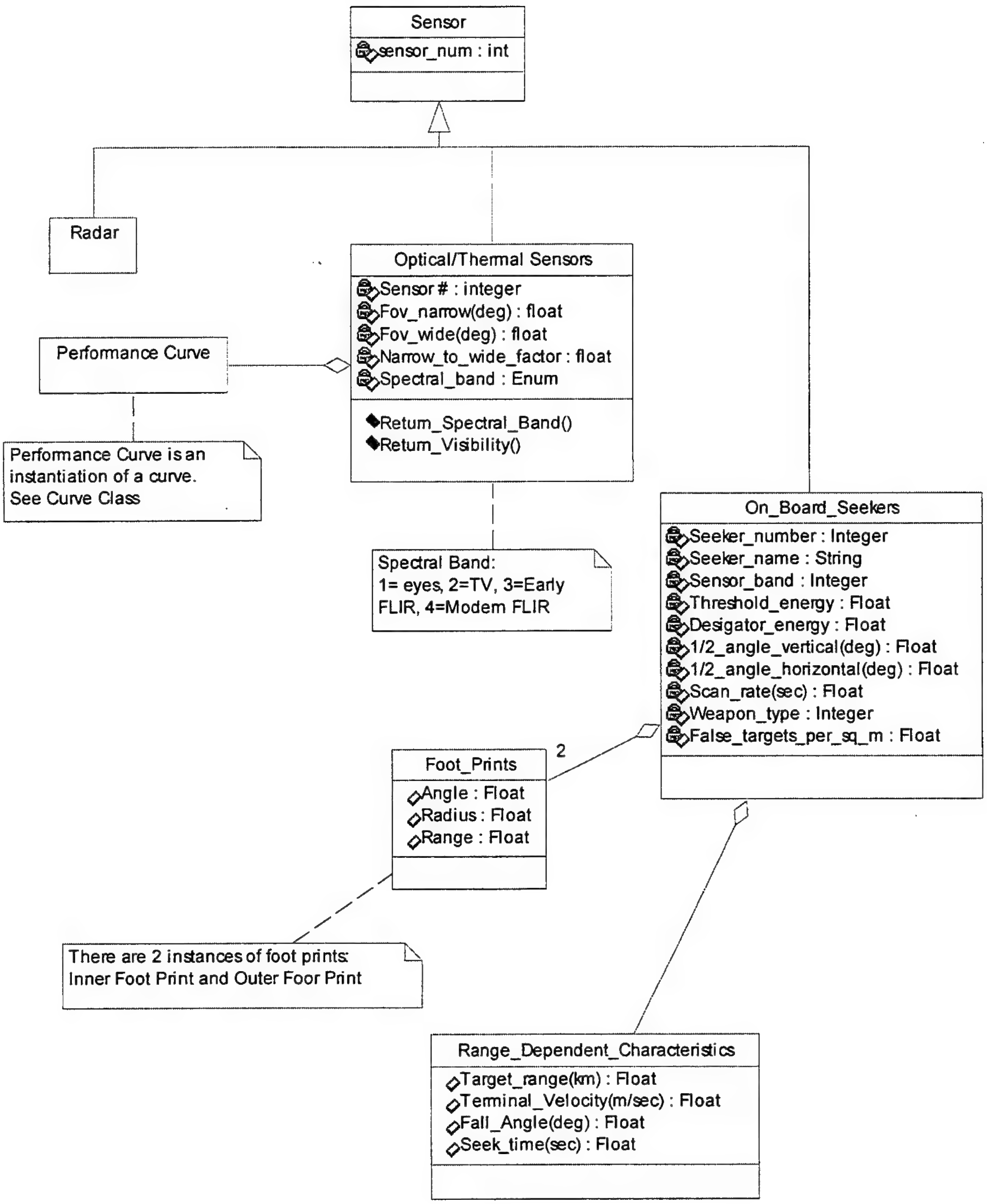
11. AIRCRAFT_UNIT AND GROUND_VEHICLE_UNIT CLASSES



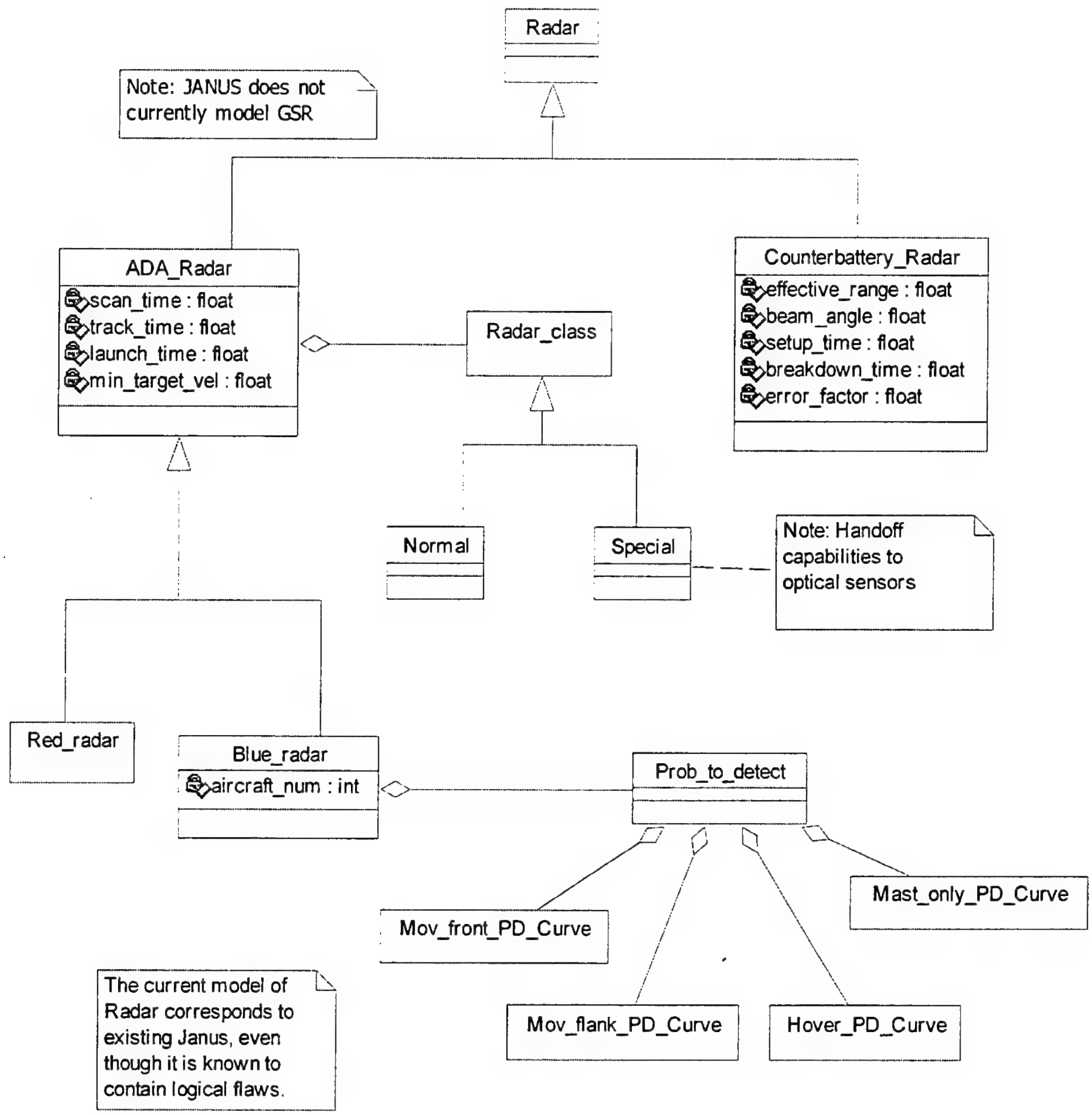
12. MOBILE_PLATFORM_SUBSYSTEM CLASS.



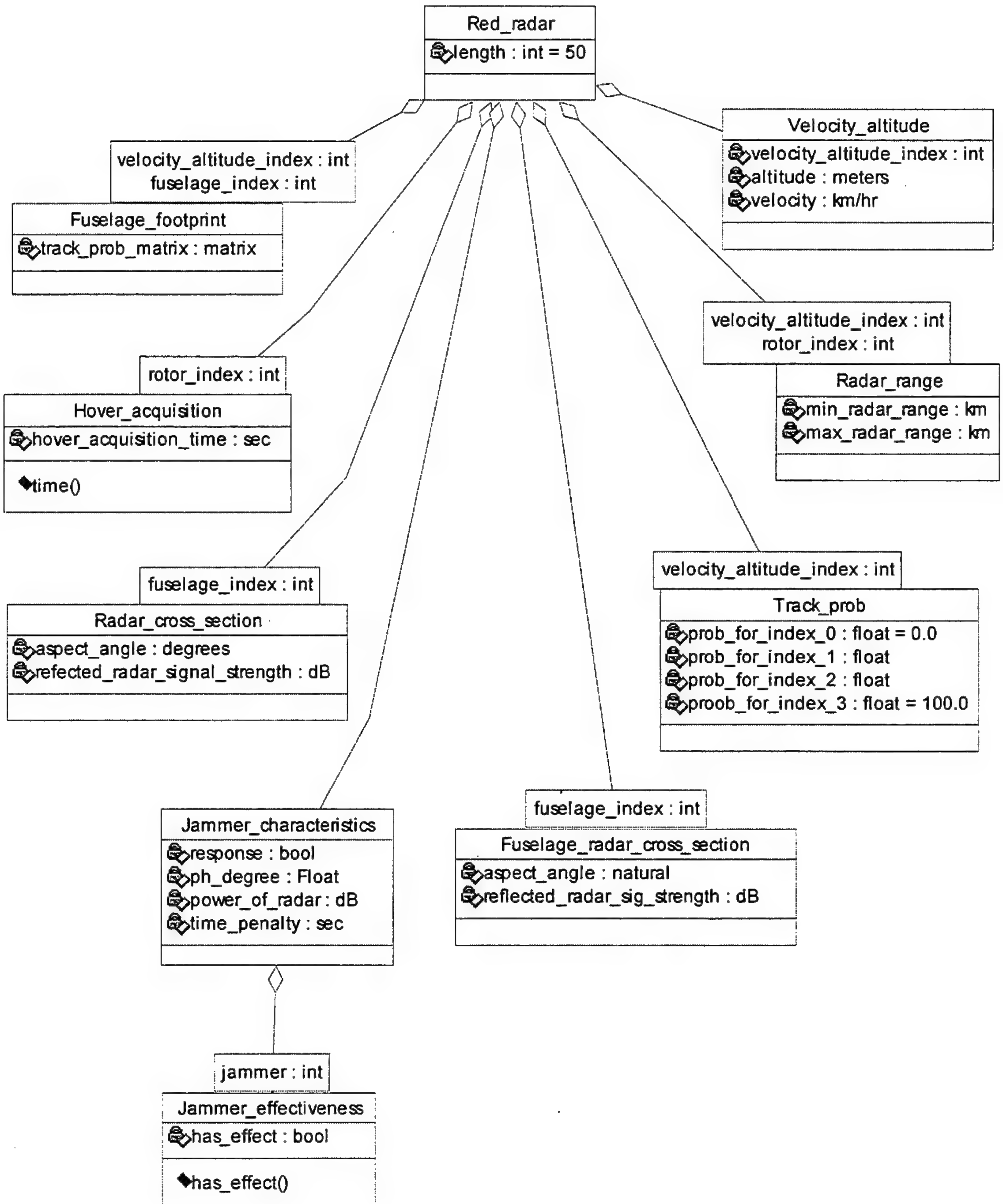
13. SENSOR CLASS



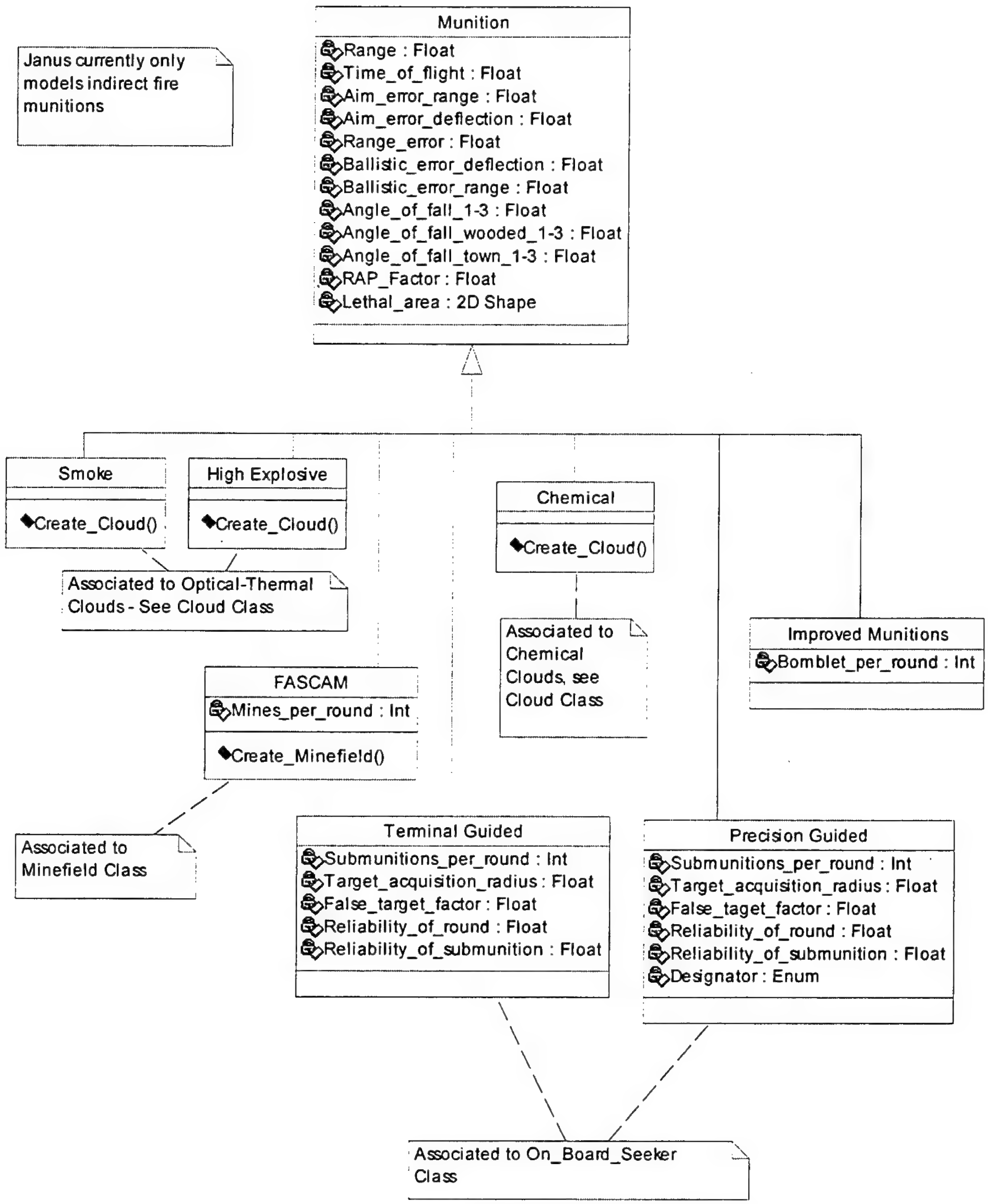
14. RADAR CLASS



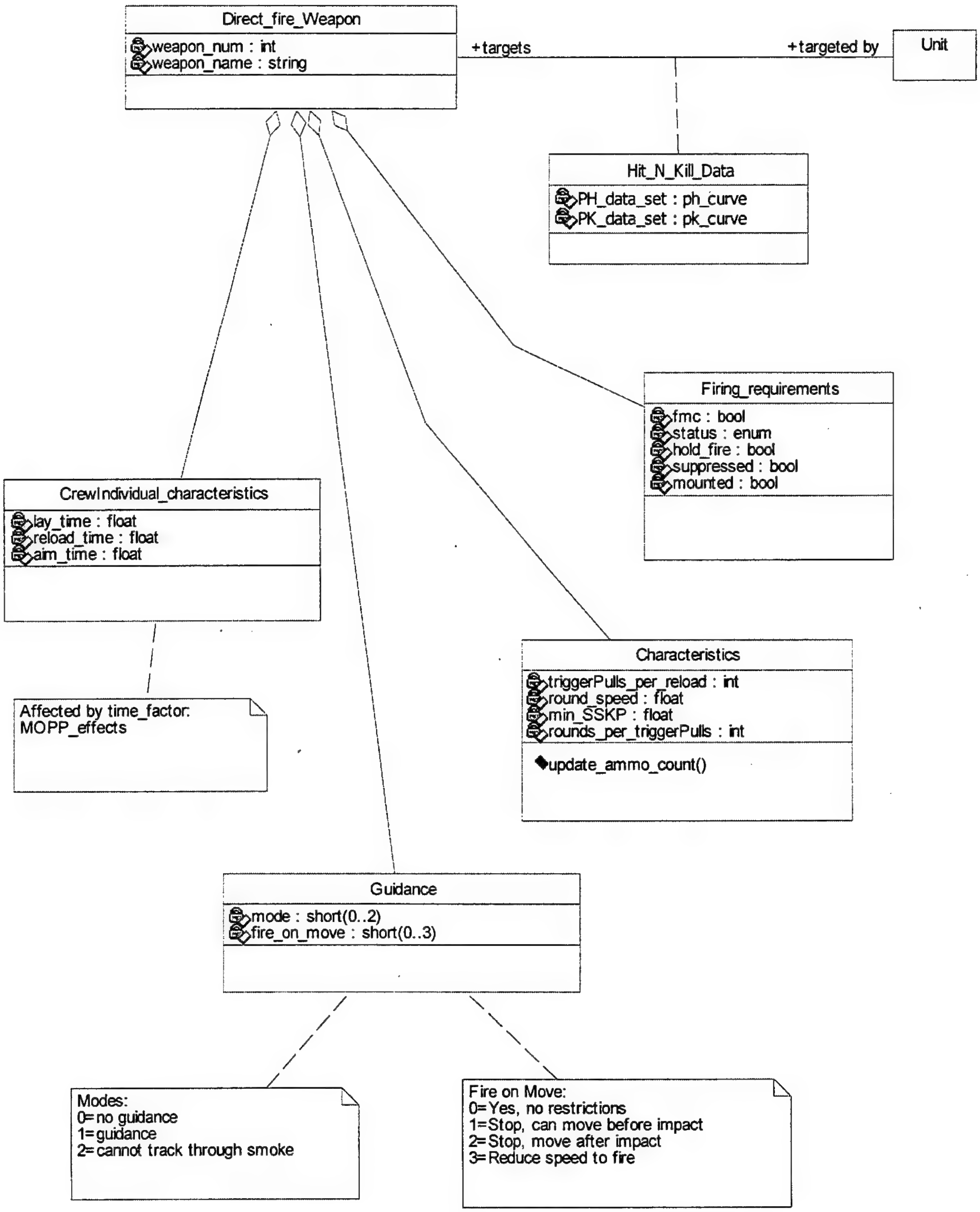
15. RED_RADAR CLASS



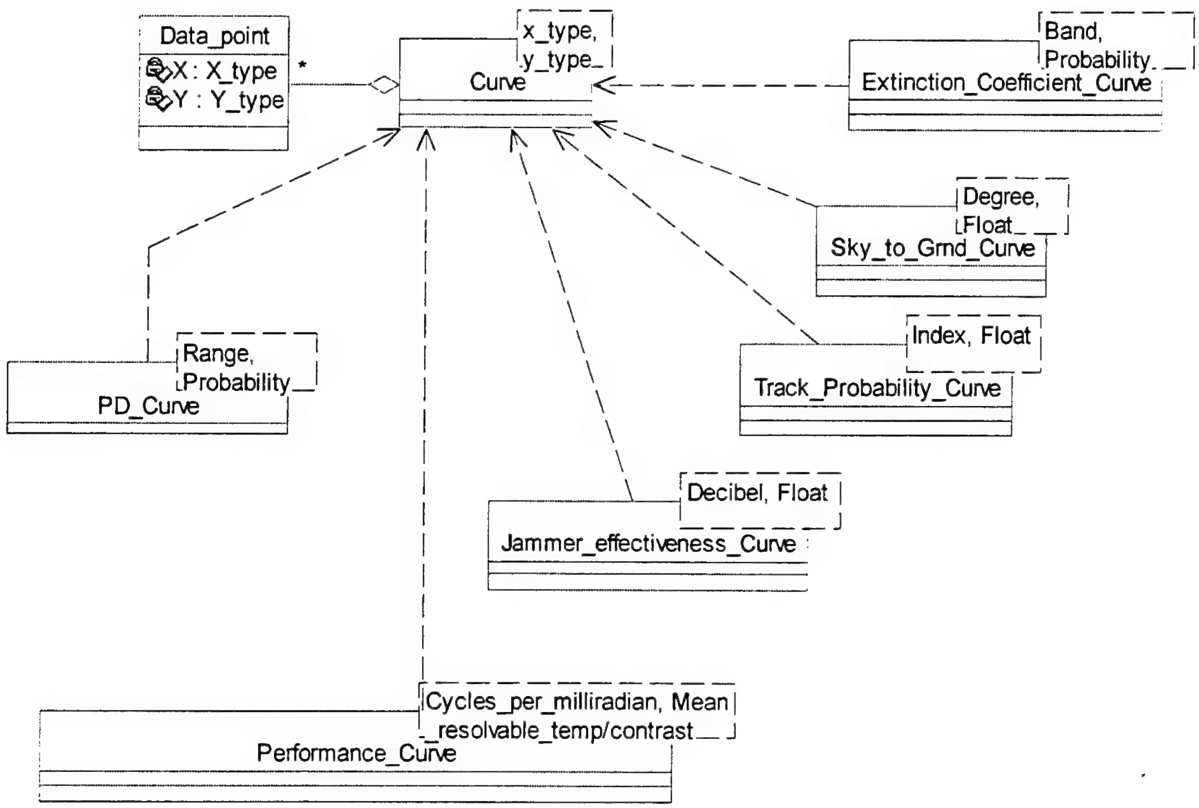
16. MUNITION_TYPE CLASS



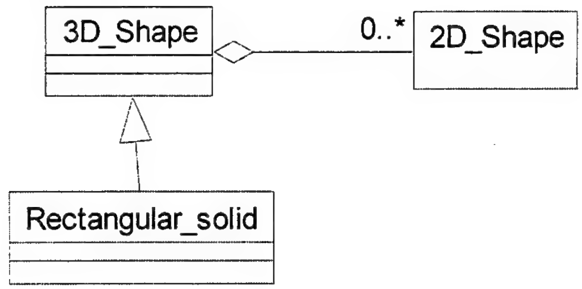
17. DIRECT_FIRE_WEAPON CLASS

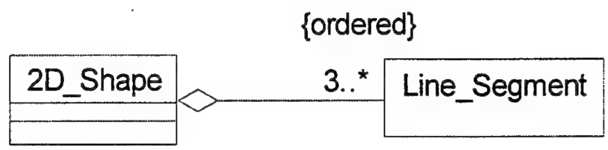


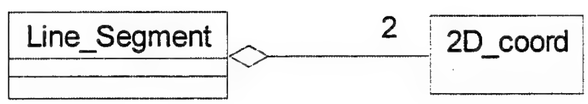
18. CURVE CLASS



19. 3D_SHAPE, 2D_SHAPE and LINE_SEGMENT CLASSES







20. PROBABILITY, 2D_COORDINATE, AND WAYPOINT DATA TYPES

```
Type Probability is 0.0..1.0
```

```
Type 2d_Coord =
   Record
     X : Float
     Y : Float
   End Record;
```

```
Type Waypoint =
   Record
     Origin : 2d_Coord
     Earliest_Time_To_Move : Float
   End Record;
```

APPENDIX F. THE PSDL SPECIFICATION FOR THE EXECUTABLE PROTOTYPE

```
TYPE event_type
SPECIFICATION
END

IMPLEMENTATION ada event_type
END

TYPE event_queue_type
SPECIFICATION

  OPERATOR empty_queue
  SPECIFICATION
    OUTPUT q: event_queue_type
  END

END

IMPLEMENTATION ada event_queue_type
END

TYPE statistics_type
SPECIFICATION
END

IMPLEMENTATION ada statistics_type
END

TYPE scenario_type
SPECIFICATION

  OPERATOR empty_scenario
  SPECIFICATION
    OUTPUT s: scenario_type
  END

END

IMPLEMENTATION ada scenario_type
END

TYPE statistics_request_type
SPECIFICATION
END

IMPLEMENTATION ada statistics_request_type
END

TYPE replay_request_type
SPECIFICATION
END

IMPLEMENTATION ada replay_request_type
END

TYPE user_interaction_type
SPECIFICATION

  OPERATOR stop_simulation
```

```
  SPECIFICATION
    OUTPUT x: user_interaction_type
  END

END

IMPLEMENTATION ada user_interaction_type
END

TYPE location_type
SPECIFICATION
END

IMPLEMENTATION ada location_type
END

TYPE game_time_type
SPECIFICATION

  OPERATOR zero
  SPECIFICATION
    OUTPUT z: game_time_type
  END

END

IMPLEMENTATION ada game_time_type
END

OPERATOR gui_3
  SPECIFICATION
    INPUT statistics: statistics_type
    INPUT replay: location_type
    OUTPUT scenario: scenario_type
    OUTPUT user_interaction: user_interaction_type
    OUTPUT replay_request: replay_request_type
    OUTPUT statistics_request: statistics_request_type
    STATES scenario: scenario_type INITIALLY scenario_type.empty_scenario
    STATES new_y: float INITIALLY 0.0
    STATES new_x: float INITIALLY 0.0
    STATES first_time: boolean INITIALLY TRUE
  END

  IMPLEMENTATION
    GRAPH
      VERTEX enter_new_plan_75_74
      VERTEX get_y_68_67
      VERTEX get_x_65_64
      VERTEX get_re_30_29
      VERTEX get_st_27_26
      VERTEX edit_plan_24_23
      VERTEX get_user_in_21_20
      VERTEX gui_event_monitor_18_17: 50 MS
      VERTEX display_st_31_30
      VERTEX display_re_37_36
      VERTEX initial_scenario_40_39
      EDGE new_plan_entered enter_new_plan_75_74 -> edit_plan_24_23
      EDGE new_y get_y_68_67 -> edit_plan_24_23
      EDGE new_x get_x_65_64 -> edit_plan_24_23
      EDGE scenario edit_plan_24_23 -> edit_plan_24_23
      EDGE scenario edit_plan_24_23 -> EXTERNAL
      EDGE statistics_request get_st_27_26 -> EXTERNAL
      EDGE replay_request get_re_30_29 -> EXTERNAL
```

```
      EDGE user_interaction get_user_in_21_20 -> EXTERNAL
      EDGE statistics EXTERNAL -> display_st_31_30
      EDGE scenario initial_scenario_40_39 -> EXTERNAL
      EDGE replay EXTERNAL -> display_re_37_36
      EDGE first_time initial_scenario_40_39 -> initial_scenario_40_39
    DATA STREAM
      new_plan_entered: boolean
    CONTROL CONSTRAINTS
      OPERATOR enter_new_plan_75_74
      OPERATOR get_y_68_67
      OPERATOR get_x_65_64
      OPERATOR get_re_30_29
      OPERATOR get_st_27_26
      OPERATOR edit_plan_24_23
        TRIGGERED BY ALL new_plan_entered
      OPERATOR get_user_in_21_20
      OPERATOR gui_event_monitor_18_17
        PERIOD 300 MS
        FINISH WITHIN 300 MS
      OPERATOR display_st_31_30
      OPERATOR display_re_37_36
      OPERATOR initial_scenario_40_39
        TRIGGERED IF (first_time = TRUE)
  END

OPERATOR warrior_1
  SPECIFICATION
    STATES replay_position: integer INITIALLY 1
    STATES replay_request: replay_request_type INITIALLY
replay_request_type.off
  END

  IMPLEMENTATION
    GRAPH
      VERTEX gui_3_2
      VERTEX post_processor_6_5
      VERTEX janus_9_8
      VERTEX jaaws_12_11
      EDGE replay_position jaaws_12_11 -> jaaws_12_11
      EDGE replay_request jaaws_12_11 -> jaaws_12_11
      EDGE scenario gui_3_2 -> janus_9_8
      EDGE user_interaction gui_3_2 -> janus_9_8
      EDGE replay_request gui_3_2 -> jaaws_12_11
      EDGE statistics_request gui_3_2 -> post_processor_6_5
      EDGE statistics post_processor_6_5 -> gui_3_2
      EDGE replay jaaws_12_11 -> gui_3_2
      EDGE simulation_history janus_9_8 -> jaaws_12_11
      EDGE simulation_history janus_9_8 -> post_processor_6_5
    DATA STREAM
      scenario: scenario_type,
      user_interaction: user_interaction_type,
      statistics_request: statistics_request_type,
      statistics: statistics_type,
      replay: location_type,
      simulation_history: sequence[e: event_type]
    CONTROL CONSTRAINTS
      OPERATOR gui_3_2
      OPERATOR post_processor_6_5
        TRIGGERED BY ALL statistics_request
      OPERATOR janus_9_8
      OPERATOR jaaws_12_11
        TRIGGERED IF (sequence.length(simulation_history) > 0)
  END
```

```
OPERATOR enter_new_plan_75
  SPECIFICATION
    OUTPUT new_plan_entered: boolean
  END

  IMPLEMENTATION tae enter_new_plan_75
  END

OPERATOR get_y_68
  SPECIFICATION
    OUTPUT new_y: float
  END

  IMPLEMENTATION tae get_y_68
  END

OPERATOR get_x_65
  SPECIFICATION
    OUTPUT new_x: float
  END

  IMPLEMENTATION tae get_x_65
  END

OPERATOR get_re_30
  SPECIFICATION
    OUTPUT replay_request: replay_request_type
  END

  IMPLEMENTATION tae get_re_30
  END

OPERATOR get_st_27
  SPECIFICATION
    OUTPUT statistics_request: statistics_request_type
  END

  IMPLEMENTATION tae get_st_27
  END

OPERATOR edit_plan_24
  SPECIFICATION
    INPUT new_plan_entered: boolean
    INPUT new_y: float
    INPUT new_x: float
    INPUT scenario: scenario_type
    OUTPUT scenario: scenario_type
  END

  IMPLEMENTATION ada edit_plan_24
  END

OPERATOR get_user_in_21
  SPECIFICATION
    OUTPUT user_interaction: user_interaction_type
  END

  IMPLEMENTATION tae get_user_in_21
  END

OPERATOR gui_event_monitor_18
  SPECIFICATION
```

```
    MAXIMUM EXECUTION TIME 50 MS
  END

  IMPLEMENTATION ada gui_event_monitor_18
  END

OPERATOR display_st_31
  SPECIFICATION
    INPUT statistics: statistics_type
  END

  IMPLEMENTATION tae display_st_31
  END

OPERATOR display_re_37
  SPECIFICATION
    INPUT replay: location_type
  END

  IMPLEMENTATION tae display_re_37
  END

OPERATOR initial_scenario_40
  SPECIFICATION
    INPUT first_time: boolean
    OUTPUT scenario: scenario_type
    OUTPUT first_time: boolean
  END

  IMPLEMENTATION ada initial_scenario_40
  END

OPERATOR post_processor_6
  SPECIFICATION
    INPUT statistics_request: statistics_request_type
    INPUT simulation_history: sequence[e: event_type]
    OUTPUT statistics: statistics_type
  END

  IMPLEMENTATION ada post_processor_6
  END

OPERATOR janus_9
  SPECIFICATION
    INPUT scenario: scenario_type
    INPUT user_interaction: user_interaction_type
    OUTPUT simulation_history: sequence[e: event_type]
    STATES game_time: game_time_type INITIALLY game_time_type.zero
    STATES event_q: event_queue_type INITIALLY event_queue_type.empty
  END

  IMPLEMENTATION
    GRAPH
      VERTEX create_new_events_114_113
      VERTEX do_event_66_65: 100 MS
      VERTEX create_user_event_69_68
      EDGE game_time do_event_66_65 -> create_user_event_69_68
      EDGE game_time do_event_66_65 -> do_event_66_65
      EDGE event_q do_event_66_65 -> do_event_66_65
      EDGE simulation_history do_event_66_65 -> do_event_66_65
      EDGE event_q create_new_events_114_113 -> do_event_66_65
      EDGE game_time do_event_66_65 -> create_new_events_114_113
      EDGE event_q do_event_66_65 -> create_new_events_114_113
```

```
      EDGE event_q create_user_event_69_68 -> do_event_66_65
      EDGE event_q do_event_66_65 -> create_user_event_69_68
      EDGE scenario EXTERNAL -> create_new_events_114_113
      EDGE simulation_history do_event_66_65 -> EXTERNAL
      EDGE user_interaction EXTERNAL -> create_user_event_69_68
    CONTROL CONSTRAINTS
      OPERATOR create_new_events_114_113
        TRIGGERED IF not(scenario_type.is_empty(scenario))
      OPERATOR do_event_66_65
        TRIGGERED IF not(event_queue_type.is_empty(event_q))
        PERIOD 1000 MS
      OPERATOR create_user_event_69_68
        TRIGGERED IF (user_interaction = stop_simulation)
  END

OPERATOR create_new_events_114
  SPECIFICATION
    INPUT game_time: game_time_type
    INPUT event_q: event_queue_type
    INPUT scenario: scenario_type
    OUTPUT event_q: event_queue_type
  END

  IMPLEMENTATION ada create_new_events_114
  END

OPERATOR do_event_66
  SPECIFICATION
    INPUT game_time: game_time_type
    INPUT simulation_history: sequence[e: event_type]
    INPUT event_q: event_queue_type
    OUTPUT game_time: game_time_type
    OUTPUT event_q: event_queue_type
    OUTPUT simulation_history: sequence[e: event_type]
    MAXIMUM EXECUTION TIME 100 MS
  END

  IMPLEMENTATION ada do_event_66
  END

OPERATOR create_user_event_69
  SPECIFICATION
    INPUT game_time: game_time_type
    INPUT event_q: event_queue_type
    INPUT user_interaction: user_interaction_type
    OUTPUT event_q: event_queue_type
  END

  IMPLEMENTATION ada create_user_event_69
  END

OPERATOR jaaws_12
  SPECIFICATION
    INPUT replay_position: integer
    INPUT replay_request: replay_request_type
    INPUT simulation_history: sequence[e: event_type]
    OUTPUT replay_position: integer
    OUTPUT replay_request: replay_request_type
    OUTPUT replay: location_type
  END

  IMPLEMENTATION ada jaaws_12
  END
```

APPENDIX G. THE ADA/C SOURCE CODE OF THE PROTOTYPE

1. WARRIOR_1.ADB

```
  with WARRIOR_1_STATIC_SCHEDULERS; use WARRIOR_1_STATIC_SCHEDULERS;
  with WARRIOR_1_DYNAMIC_SCHEDULERS; use WARRIOR_1_DYNAMIC_SCHEDULERS;
  with CAPS_HARDWARE_MODEL; use CAPS_HARDWARE_MODEL;
procedure WARRIOR_1 is
begin
  init_hardware_model;
  start_static_schedule;
  start_dynamic_schedule;
end WARRIOR_1;
```

2. WARRIOR_1_DRIVERS.ADS

```
package WARRIOR_1_DRIVERS is
  procedure POST_PROCESSOR_6_5_DRIVER;
  procedure JAAWS_12_11_DRIVER;
  procedure ENTER_NEW_PLAN_75_74_DRIVER;
  procedure GET_Y_68_67_DRIVER;
  procedure GET_X_65_64_DRIVER;
  procedure GET_RE_30_29_DRIVER;
  procedure GET_ST_27_26_DRIVER;
  procedure EDIT_PLAN_24_23_DRIVER;
  procedure GET_USER_IN_21_20_DRIVER;
  procedure GUI_EVENT_MONITOR_18_17_DRIVER;
  procedure DISPLAY_ST_31_30_DRIVER;
  procedure DISPLAY_RE_37_36_DRIVER;
  procedure INITIAL_SCENARIO_40_39_DRIVER;
  procedure CREATE_NEW_EVENTS_114_113_DRIVER;
  procedure DO_EVENT_66_65_DRIVER;
  procedure CREATE_USER_EVENT_69_68_DRIVER;
end WARRIOR_1_DRIVERS;
```

3. WARRIOR_1_DRIVERS.ADB

```
  -- with/use clauses for atomic components.
  with EVENT_TYPE_PKG; use EVENT_TYPE_PKG;
  with EVENT_QUEUE_TYPE_PKG; use EVENT_QUEUE_TYPE_PKG;
  with STATISTICS_TYPE_PKG; use STATISTICS_TYPE_PKG;
  with SCENARIO_TYPE_PKG; use SCENARIO_TYPE_PKG;
  with STATISTICS_REQUEST_TYPE_PKG; use STATISTICS_REQUEST_TYPE_PKG;
  with REPLAY_REQUEST_TYPE_PKG; use REPLAY_REQUEST_TYPE_PKG;
  with USER_INTERACTION_TYPE_PKG; use USER_INTERACTION_TYPE_PKG;
  with LOCATION_TYPE_PKG; use LOCATION_TYPE_PKG;
  with GAME_TIME_TYPE_PKG; use GAME_TIME_TYPE_PKG;
  with ENTER_NEW_PLAN_75_PKG; use ENTER_NEW_PLAN_75_PKG;
  with GET_Y_68_PKG; use GET_Y_68_PKG;
  with GET_X_65_PKG; use GET_X_65_PKG;
  with GET_RE_30_PKG; use GET_RE_30_PKG;
  with GET_ST_27_PKG; use GET_ST_27_PKG;
  with EDIT_PLAN_24_PKG; use EDIT_PLAN_24_PKG;
  with GET_USER_IN_21_PKG; use GET_USER_IN_21_PKG;
  with GUI_EVENT_MONITOR_18_PKG; use GUI_EVENT_MONITOR_18_PKG;
  with DISPLAY_ST_31_PKG; use DISPLAY_ST_31_PKG;
  with DISPLAY_RE_37_PKG; use DISPLAY_RE_37_PKG;
  with INITIAL_SCENARIO_40_PKG; use INITIAL_SCENARIO_40_PKG;
  with POST_PROCESSOR_6_PKG; use POST_PROCESSOR_6_PKG;
  with CREATE_NEW_EVENTS_114_PKG; use CREATE_NEW_EVENTS_114_PKG;
```

```
  with DO_EVENT_66_PKG; use DO_EVENT_66_PKG;
  with CREATE_USER_EVENT_69_PKG; use CREATE_USER_EVENT_69_PKG;
  with JAAWS_12_PKG; use JAAWS_12_PKG;
  -- with/use clauses for generated packages.
  with WARRIOR_1_EXCEPTIONS; use WARRIOR_1_EXCEPTIONS;
  with WARRIOR_1_STREAMS; use WARRIOR_1_STREAMS;
  with WARRIOR_1_TIMERS; use WARRIOR_1_TIMERS;
  with WARRIOR_1_INSTANTIATIONS; use WARRIOR_1_INSTANTIATIONS;
  -- with/use clauses for CAPS library packages.
  with DS_DEBUG_PKG; use DS_DEBUG_PKG;
  with PSDL_STREAMS; use PSDL_STREAMS;
  with PSDL_STRING_PKG; use PSDL_STRING_PKG;
  with PSDL_TIMERS;
package body WARRIOR_1_DRIVERS is

  procedure POST_PROCESSOR_6_5_DRIVER is
    LV_STATISTICS_REQUEST :
            STATISTICS_REQUEST_TYPE_PKG.STATISTICS_REQUEST_TYPE;
    LV_SIMULATION_HISTORY : EVENT_TYPE_SEQUENCE;
    LV_STATISTICS : STATISTICS_TYPE_PKG.STATISTICS_TYPE;

    EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
    EXCEPTION_ID: PSDL_EXCEPTION;
  begin
    -- Data trigger checks.
    if not (DS_STATISTICS_REQUEST_POST_PROCESSOR_6_5.NEW_DATA) then
      return;
    end if;

    -- Data stream reads.
    begin
      DS_STATISTICS_REQUEST_POST_PROCESSOR_6_5.BUFFER.READ(
                                               LV_STATISTICS_REQUEST);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("STATISTICS_REQUEST_POST_PROCESSOR_6_5",
                                              "POST_PROCESSOR_6_5");
    end;
    begin
      DS_SIMULATION_HISTORY_POST_PROCESSOR_6_5.BUFFER.READ(
                                               LV_SIMULATION_HISTORY);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("SIMULATION_HISTORY_POST_PROCESSOR_6_5",
                                              "POST_PROCESSOR_6_5");
    end;

    -- Execution trigger condition check.
    if True then
      begin
      POST_PROCESSOR_6(
        STATISTICS_REQUEST => LV_STATISTICS_REQUEST,
        SIMULATION_HISTORY => LV_SIMULATION_HISTORY,
        STATISTICS => LV_STATISTICS);
      exception
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("POST_PROCESSOR_6_5");
          EXCEPTION_HAS_OCCURRED := true;
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
      end;
    else return;
    end if;
```

```
-- Exception Constraint translations.

-- Other constraint option translations.

-- Unconditional output translations.
  if not EXCEPTION_HAS_OCCURRED then
    begin
      DS_STATISTICS_DISPLAY_ST_31_30.BUFFER.WRITE(LV_STATISTICS);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("STATISTICS_DISPLAY_ST_31_30",
                                                   "POST_PROCESSOR_6_5");
    end;
  end if;

-- PSDL Exception handler.
  if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
      "POST_PROCESSOR_6_5",
      PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
  end if;
end POST_PROCESSOR_6_5_DRIVER;


procedure JAAWS_12_11_DRIVER is
  LV_SIMULATION_HISTORY : EVENT_TYPE_SEQUENCE;
  LV_REPLAY_POSITION : INTEGER;
  LV_REPLAY_REQUEST : REPLAY_REQUEST_TYPE_PKG.REPLAY_REQUEST_TYPE;
  LV_REPLAY : LOCATION_TYPE_PKG.LOCATION_TYPE;

  EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
-- Data trigger checks.

-- Data stream reads.
  begin
    DS_REPLAY_POSITION_JAAWS_12_11.BUFFER.READ(LV_REPLAY_POSITION);
  exception
    when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("REPLAY_POSITION_JAAWS_12_11",
                                                   "JAAWS_12_11");
  end;
  begin
    DS_REPLAY_REQUEST_JAAWS_12_11.BUFFER.READ(LV_REPLAY_REQUEST);
  exception
    when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("REPLAY_REQUEST_JAAWS_12_11",
                                                   "JAAWS_12_11");
  end;
  begin
    DS_SIMULATION_HISTORY_JAAWS_12_11.BUFFER.READ(LV_SIMULATION_HISTORY);
  exception
    when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("SIMULATION_HISTORY_JAAWS_12_11",
                                                   "JAAWS_12_11");
  end;

-- Execution trigger condition check.
  if  (LENGTH(LV_SIMULATION_HISTORY) > 0)  then
    begin
    JAAWS_12(
      SIMULATION_HISTORY => LV_SIMULATION_HISTORY,
```

```
      REPLAY_POSITION => LV_REPLAY_POSITION,
      REPLAY_REQUEST => LV_REPLAY_REQUEST,
      REPLAY => LV_REPLAY);
    exception
      when others =>
        DS_DEBUG.UNDECLARED_EXCEPTION("JAAWS_12_11");
        EXCEPTION_HAS_OCCURRED := true;
        EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
  else return;
  end if;

-- Exception Constraint translations.

-- Other constraint option translations.

-- Unconditional output translations.
  if not EXCEPTION_HAS_OCCURRED then
    begin
      DS_REPLAY_POSITION_JAAWS_12_11.BUFFER.WRITE(LV_REPLAY_POSITION);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("REPLAY_POSITION_JAAWS_12_11",
                                                  "JAAWS_12_11");
    end;
  end if;
  if not EXCEPTION_HAS_OCCURRED then
    begin
      DS_REPLAY_REQUEST_JAAWS_12_11.BUFFER.WRITE(LV_REPLAY_REQUEST);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("REPLAY_REQUEST_JAAWS_12_11",
                                                  "JAAWS_12_11");
    end;
  end if;
  if not EXCEPTION_HAS_OCCURRED then
    begin
      DS_REPLAY_DISPLAY_RE_37_36.BUFFER.WRITE(LV_REPLAY);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("REPLAY_DISPLAY_RE_37_36", "JAAWS_12_11");
    end;
  end if;

-- PSDL Exception handler.
  if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
      "JAAWS_12_11",
      PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
  end if;
end JAAWS_12_11_DRIVER;


procedure ENTER_NEW_PLAN_75_74_DRIVER is
  LV_NEW_PLAN_ENTERED : BOOLEAN;

  EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
-- Data trigger checks.

-- Data stream reads.
```

```
-- Execution trigger condition check.
  if ENTER_NEW_PLAN_75_PKG.has_new_input then
    begin
    ENTER_NEW_PLAN_75(
      NEW_PLAN_ENTERED => LV_NEW_PLAN_ENTERED);
    exception
      when others =>
        DS_DEBUG.UNDECLARED_EXCEPTION("ENTER_NEW_PLAN_75_74");
        EXCEPTION_HAS_OCCURRED := true;
        EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
  else return;
  end if;

-- Exception Constraint translations.

-- Other constraint option translations.

-- Unconditional output translations.
  if not EXCEPTION_HAS_OCCURRED then
    begin
      DS_NEW_PLAN_ENTERED_EDIT_PLAN_24_23.BUFFER.WRITE(
                                                  LV_NEW_PLAN_ENTERED);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("NEW_PLAN_ENTERED_EDIT_PLAN_24_23",
                                                  "ENTER_NEW_PLAN_75_74");
    end;
  end if;

-- PSDL Exception handler.
  if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
      "ENTER_NEW_PLAN_75_74",
      PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
  end if;
end ENTER_NEW_PLAN_75_74_DRIVER;


procedure GET_Y_68_67_DRIVER is
  LV_NEW_Y : FLOAT;

  EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
   -- Data trigger checks.

   -- Data stream reads.

   -- Execution trigger condition check.
  if GET_Y_68_PKG.has_new_input then
    begin
    GET_Y_68(
      NEW_Y => LV_NEW_Y);
    exception
      when others =>
        DS_DEBUG.UNDECLARED_EXCEPTION("GET_Y_68_67");
        EXCEPTION_HAS_OCCURRED := true;
        EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
  else return;
  end if;
```

```
-- Exception Constraint translations.

-- Other constraint option translations.

-- Unconditional output translations.
  if not EXCEPTION_HAS_OCCURRED then
    begin
      DS_NEW_Y_EDIT_PLAN_24_23.BUFFER.WRITE(LV_NEW_Y);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("NEW_Y_EDIT_PLAN_24_23", "GET_Y_68_67");
    end;
  end if;

-- PSDL Exception handler.
  if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
      "GET_Y_68_67",
      PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
  end if;
end GET_Y_68_67_DRIVER;


procedure GET_X_65_64_DRIVER is
  LV_NEW_X : FLOAT;

  EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
-- Data trigger checks.

-- Data stream reads.

-- Execution trigger condition check.
  if GET_X_65_PKG.has_new_input then
    begin
    GET_X_65(
      NEW_X => LV_NEW_X);
    exception
      when others =>
        DS_DEBUG.UNDECLARED_EXCEPTION("GET_X_65_64");
        EXCEPTION_HAS_OCCURRED := true;
        EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
  else return;
  end if;

-- Exception Constraint translations.

-- Other constraint option translations.

-- Unconditional output translations.
  if not EXCEPTION_HAS_OCCURRED then
    begin
      DS_NEW_X_EDIT_PLAN_24_23.BUFFER.WRITE(LV_NEW_X);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("NEW_X_EDIT_PLAN_24_23", "GET_X_65_64");
    end;
  end if;

-- PSDL Exception handler.
  if EXCEPTION_HAS_OCCURRED then
```

```
    DS_DEBUG.UNHANDLED_EXCEPTION(
      "GET_X_65_64",
      PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
  end if;
end GET_X_65_64_DRIVER;


procedure GET_RE_30_29_DRIVER is
  LV_REPLAY_REQUEST : REPLAY_REQUEST_TYPE_PKG.REPLAY_REQUEST_TYPE;

  EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
-- Data trigger checks.

-- Data stream reads.

-- Execution trigger condition check.
  if GET_RE_30_PKG.has_new_input then
    begin
    GET_RE_30(
      REPLAY_REQUEST => LV_REPLAY_REQUEST);
    exception
      when others =>
        DS_DEBUG.UNDECLARED_EXCEPTION("GET_RE_30_29");
        EXCEPTION_HAS_OCCURRED := true;
        EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
  else return;
  end if;

-- Exception Constraint translations.

-- Other constraint option translations.

-- Unconditional output translations.
  if not EXCEPTION_HAS_OCCURRED then
    begin
      DS_REPLAY_REQUEST_JAAWS_12_11.BUFFER.WRITE(LV_REPLAY_REQUEST);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("REPLAY_REQUEST_JAAWS_12_11",
                                                          "GET_RE_30_29");
    end;
  end if;

-- PSDL Exception handler.
  if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
      "GET_RE_30_29",
      PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
  end if;
end GET_RE_30_29_DRIVER;


procedure GET_ST_27_26_DRIVER is
  LV_STATISTICS_REQUEST :
                      STATISTICS_REQUEST_TYPE_PKG.STATISTICS_REQUEST_TYPE;

  EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
-- Data trigger checks.
```

```
-- Data stream reads.

-- Execution trigger condition check.
   if GET_ST_27_PKG.has_new_input then
     begin
     GET_ST_27(
       STATISTICS_REQUEST => LV_STATISTICS_REQUEST);
     exception
       when others =>
         DS_DEBUG.UNDECLARED_EXCEPTION("GET_ST_27_26");
         EXCEPTION_HAS_OCCURRED := true;
         EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
     end;
   else return;
   end if;

 -- Exception Constraint translations.

 -- Other constraint option translations.

 -- Unconditional output translations.
   if not EXCEPTION_HAS_OCCURRED then
     begin
       DS_STATISTICS_REQUEST_POST_PROCESSOR_6_5.BUFFER.WRITE(
                                           LV_STATISTICS_REQUEST);
     exception
       when BUFFER_OVERFLOW =>
         DS_DEBUG.BUFFER_OVERFLOW("STATISTICS_REQUEST_POST_PROCESSOR_6_5",
                                                     "GET_ST_27_26");
     end;
   end if;

-- PSDL Exception handler.
   if EXCEPTION_HAS_OCCURRED then
     DS_DEBUG.UNHANDLED_EXCEPTION(
       "GET_ST_27_26",
       PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
   end if;
end GET_ST_27_26_DRIVER;


procedure EDIT_PLAN_24_23_DRIVER is
  LV_NEW_PLAN_ENTERED : BOOLEAN;
  LV_NEW_Y : FLOAT;
  LV_NEW_X : FLOAT;
  LV_SCENARIO : SCENARIO_TYPE_PKG.SCENARIO_TYPE;

  EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
-- Data trigger checks.
  if not (DS_NEW_PLAN_ENTERED_EDIT_PLAN_24_23.NEW_DATA) then
    return;
  end if;

-- Data stream reads.
  begin
    DS_NEW_PLAN_ENTERED_EDIT_PLAN_24_23.BUFFER.READ(LV_NEW_PLAN_ENTERED);
  exception
    when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("NEW_PLAN_ENTERED_EDIT_PLAN_24_23",
```

```
                                                        "EDIT_PLAN_24_23");
  end;
  begin
    DS_NEW_Y_EDIT_PLAN_24_23.BUFFER.READ(LV_NEW_Y);
  exception
    when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("NEW_Y_EDIT_PLAN_24_23",
                                                        "EDIT_PLAN_24_23");
  end;
  begin
    DS_NEW_X_EDIT_PLAN_24_23.BUFFER.READ(LV_NEW_X);
  exception
    when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("NEW_X_EDIT_PLAN_24_23",
                                                        "EDIT_PLAN_24_23");
  end;
  begin
    DS_SCENARIO_EDIT_PLAN_24_23.BUFFER.READ(LV_SCENARIO);
  exception
    when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("SCENARIO_EDIT_PLAN_24_23",
                                                        "EDIT_PLAN_24_23");
  end;

-- Execution trigger condition check.
  if True then
    begin
    EDIT_PLAN_24(
      NEW_PLAN_ENTERED => LV_NEW_PLAN_ENTERED,
      NEW_Y => LV_NEW_Y,
      NEW_X => LV_NEW_X,
      SCENARIO => LV_SCENARIO);
    exception
      when others =>
        DS_DEBUG.UNDECLARED_EXCEPTION("EDIT_PLAN_24_23");
        EXCEPTION_HAS_OCCURRED := true;
        EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
  else return;
  end if;

-- Exception Constraint translations.

-- Other constraint option translations.

-- Unconditional output translations.
  if not EXCEPTION_HAS_OCCURRED then
    begin
      DS_SCENARIO_CREATE_NEW_EVENTS_114_113.BUFFER.WRITE(LV_SCENARIO);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("SCENARIO_CREATE_NEW_EVENTS_114_113",

                                                               "EDIT_PLAN_24_23");
    end;
    begin
      DS_SCENARIO_EDIT_PLAN_24_23.BUFFER.WRITE(LV_SCENARIO);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("SCENARIO_EDIT_PLAN_24_23",
                                                        "EDIT_PLAN_24_23");
    end;
```

```
  end if;

-- PSDL Exception handler.
  if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
      "EDIT_PLAN_24_23",
      PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
  end if;
end EDIT_PLAN_24_23_DRIVER;


procedure GET_USER_IN_21_20_DRIVER is
  LV_USER_INTERACTION : USER_INTERACTION_TYPE_PKG.USER_INTERACTION_TYPE;

  EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
-- Data trigger checks.

-- Data stream reads.

-- Execution trigger condition check.
  if GET_USER_IN_21_PKG.has_new_input then
    begin
    GET_USER_IN_21(
      USER_INTERACTION => LV_USER_INTERACTION);
    exception
      when others =>
        DS_DEBUG.UNDECLARED_EXCEPTION("GET_USER_IN_21_20");
        EXCEPTION_HAS_OCCURRED := true;
        EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
  else return;
  end if;

-- Exception Constraint translations.

-- Other constraint option translations.

-- Unconditional output translations.
  if not EXCEPTION_HAS_OCCURRED then
    begin
      DS_USER_INTERACTION_CREATE_USER_EVENT_69_68.BUFFER.WRITE(
                                                   LV_USER_INTERACTION);
    exception
       when BUFFER_OVERFLOW =>
       DS_DEBUG.BUFFER_OVERFLOW(
         "USER_INTERACTION_CREATE_USER_EVENT_69_68", "GET_USER_IN_21_20");
    end;
  end if;

-- PSDL Exception handler.
  if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
      "GET_USER_IN_21_20",
      PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
  end if;
end GET_USER_IN_21_20_DRIVER;


procedure GUI_EVENT_MONITOR_18_17_DRIVER is

  EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
```

```
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
-- Data trigger checks.

-- Data stream reads.

-- Execution trigger condition check.
  if True then
    begin
    GUI_EVENT_MONITOR_18;
    exception
      when others =>
        DS_DEBUG.UNDECLARED_EXCEPTION("GUI_EVENT_MONITOR_18_17");
        EXCEPTION_HAS_OCCURRED := true;
        EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
  else return;
  end if;

-- Exception Constraint translations.

-- Other constraint option translations.

-- Unconditional output translations.

-- PSDL Exception handler.
  if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
      "GUI_EVENT_MONITOR_18_17",
      PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
  end if;
end GUI_EVENT_MONITOR_18_17_DRIVER;


procedure DISPLAY_ST_31_30_DRIVER is
  LV_STATISTICS : STATISTICS_TYPE_PKG.STATISTICS_TYPE;

  EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
-- Data trigger checks.

-- Data stream reads.
  begin
    DS_STATISTICS_DISPLAY_ST_31_30.BUFFER.READ(LV_STATISTICS);
  exception
    when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("STATISTICS_DISPLAY_ST_31_30",
                                                  "DISPLAY_ST_31_30");
  end;

-- Execution trigger condition check.
  if True then
    begin
    DISPLAY_ST_31(
      STATISTICS => LV_STATISTICS);
    exception
      when others =>
        DS_DEBUG.UNDECLARED_EXCEPTION("DISPLAY_ST_31_30");
        EXCEPTION_HAS_OCCURRED := true;
        EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
  else return;
```

```
  end if;

-- Exception Constraint translations.

-- Other constraint option translations.

-- Unconditional output translations.

-- PSDL Exception handler.
  if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
      "DISPLAY_ST_31_30",
      PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
  end if;
end DISPLAY_ST_31_30_DRIVER;


procedure DISPLAY_RE_37_36_DRIVER is
  LV_REPLAY : LOCATION_TYPE_PKG.LOCATION_TYPE;

  EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
-- Data trigger checks.

-- Data stream reads.
  begin
    DS_REPLAY_DISPLAY_RE_37_36.BUFFER.READ(LV_REPLAY);
  exception
    when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("REPLAY_DISPLAY_RE_37_36",
                                                     "DISPLAY_RE_37_36");
  end;

-- Execution trigger condition check.
  if True then
    begin
    DISPLAY_RE_37(
      REPLAY => LV_REPLAY);
    exception
      when others =>
        DS_DEBUG.UNDECLARED_EXCEPTION("DISPLAY_RE_37_36");
        EXCEPTION_HAS_OCCURRED := true;
        EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
  else return;
  end if;

-- Exception Constraint translations.

-- Other constraint option translations.

-- Unconditional output translations.

-- PSDL Exception handler.
  if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
      "DISPLAY_RE_37_36",
      PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
  end if;
end DISPLAY_RE_37_36_DRIVER;
```

```
procedure INITIAL_SCENARIO_40_39_DRIVER is
  LV_SCENARIO : SCENARIO_TYPE_PKG.SCENARIO_TYPE;
  LV_FIRST_TIME : BOOLEAN;

  EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
-- Data trigger checks.

-- Data stream reads.
  begin
    DS_FIRST_TIME_INITIAL_SCENARIO_40_39.BUFFER.READ(LV_FIRST_TIME);
  exception
    when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("FIRST_TIME_INITIAL_SCENARIO_40_39",
                                                     "INITIAL_SCENARIO_40_39");
  end;

-- Execution trigger condition check.
  if  (LV_FIRST_TIME = true)  then
    begin
    INITIAL_SCENARIO_40(
      SCENARIO => LV_SCENARIO,
      FIRST_TIME => LV_FIRST_TIME);
    exception
      when others =>
        DS_DEBUG.UNDECLARED_EXCEPTION("INITIAL_SCENARIO_40_39");
        EXCEPTION_HAS_OCCURRED := true;
        EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
  else return;
  end if;

 -- Exception Constraint translations.

 -- Other constraint option translations.

 - Unconditional output translations.
  if not EXCEPTION_HAS_OCCURRED then
    begin
      DS_SCENARIO_CREATE_NEW_EVENTS_114_113.BUFFER.WRITE(LV_SCENARIO);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("SCENARIO_CREATE_NEW_EVENTS_114_113",
                                                     "INITIAL_SCENARIO_40_39");
    end;
    begin
      DS_SCENARIO_EDIT_PLAN_24_23.BUFFER.WRITE(LV_SCENARIO);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("SCENARIO_EDIT_PLAN_24_23",
                                                     "INITIAL_SCENARIO_40_39");
    end;
  end if;
  if not EXCEPTION_HAS_OCCURRED then
    begin
      DS_FIRST_TIME_INITIAL_SCENARIO_40_39.BUFFER.WRITE(LV_FIRST_TIME);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("FIRST_TIME_INITIAL_SCENARIO_40_39",
                                                     "INITIAL_SCENARIO_40_39");
    end;
  end if;
```

```
-- PSDL Exception handler.
  if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
      "INITIAL_SCENARIO_40_39",
      PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
  end if;
end INITIAL_SCENARIO_40_39_DRIVER;


procedure CREATE_NEW_EVENTS_114_113_DRIVER is
  LV_GAME_TIME : GAME_TIME_TYPE_PKG.GAME_TIME_TYPE;
  LV_SCENARIO : SCENARIO_TYPE_PKG.SCENARIO_TYPE;
  LV_EVENT_Q : EVENT_QUEUE_TYPE_PKG.EVENT_QUEUE_TYPE;

  EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
-- Data trigger checks.

-- Data stream reads.
  begin
    DS_GAME_TIME_CREATE_NEW_EVENTS_114_113.BUFFER.READ(LV_GAME_TIME);
  exception
    when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("GAME_TIME_CREATE_NEW_EVENTS_114_113",
                                "CREATE_NEW_EVENTS_114_113");
  end;
  begin
    DS_EVENT_Q_CREATE_NEW_EVENTS_114_113.BUFFER.READ(LV_EVENT_Q);
  exception
    when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("EVENT_Q_CREATE_NEW_EVENTS_114_113",
                                "CREATE_NEW_EVENTS_114_113");
  end;
  begin
    DS_SCENARIO_CREATE_NEW_EVENTS_114_113.BUFFER.READ(LV_SCENARIO);
  exception
    when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("SCENARIO_CREATE_NEW_EVENTS_114_113",
                                "CREATE_NEW_EVENTS_114_113");
  end;

-- Execution trigger condition check.
  if not (SCENARIO_TYPE_PKG.IS_EMPTY(LV_SCENARIO))  then
    begin
    CREATE_NEW_EVENTS_114(
      GAME_TIME => LV_GAME_TIME,
      SCENARIO => LV_SCENARIO,
      EVENT_Q => LV_EVENT_Q);
    exception
      when others =>
        DS_DEBUG.UNDECLARED_EXCEPTION("CREATE_NEW_EVENTS_114_113");
        EXCEPTION_HAS_OCCURRED := true;
        EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
    end;
  else return;
  end if;

-- Exception Constraint translations.

-- Other constraint option translations.
```

```
-- Unconditional output translations.
  if not EXCEPTION_HAS_OCCURRED then
    begin
      DS_EVENT_Q_CREATE_USER_EVENT_69_68.BUFFER.WRITE(LV_EVENT_Q);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("EVENT_Q_CREATE_USER_EVENT_69_68",
                                    "CREATE_NEW_EVENTS_114_113");
    end;
    begin
      DS_EVENT_Q_CREATE_NEW_EVENTS_114_113.BUFFER.WRITE(LV_EVENT_Q);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("EVENT_Q_CREATE_NEW_EVENTS_114_113",
                                    "CREATE_NEW_EVENTS_114_113");
    end;
    begin
      DS_EVENT_Q_DO_EVENT_66_65.BUFFER.WRITE(LV_EVENT_Q);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("EVENT_Q_DO_EVENT_66_65",
                                    "CREATE_NEW_EVENTS_114_113");
    end;
  end if;

-- PSDL Exception handler.
  if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
      "CREATE_NEW_EVENTS_114_113",
      PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
  end if;
end CREATE_NEW_EVENTS_114_113_DRIVER;


procedure DO_EVENT_66_65_DRIVER is
  LV_GAME_TIME : GAME_TIME_TYPE_PKG.GAME_TIME_TYPE;
  LV_EVENT_Q : EVENT_QUEUE_TYPE_PKG.EVENT_QUEUE_TYPE;
  LV_SIMULATION_HISTORY : EVENT_TYPE_SEQUENCE;

  EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
-- Data trigger checks.

-- Data stream reads.
  begin
    DS_GAME_TIME_DO_EVENT_66_65.BUFFER.READ(LV_GAME_TIME);
  exception
    when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("GAME_TIME_DO_EVENT_66_65",
                                   "DO_EVENT_66_65");
  end;
  begin
    DS_SIMULATION_HISTORY_DO_EVENT_66_65.BUFFER.READ(
                                   LV_SIMULATION_HISTORY);
  exception
    when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("SIMULATION_HISTORY_DO_EVENT_66_65",
                                       "DO_EVENT_66_65");
  end;
  begin
    DS_EVENT_Q_DO_EVENT_66_65.BUFFER.READ(LV_EVENT_Q);
  exception
```

```
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("EVENT_Q_DO_EVENT_66_65",
                                                    "DO_EVENT_66_65");
    end;

 -- Execution trigger condition check.
    if not (EVENT_QUEUE_TYPE_PKG.IS_EMPTY(LV_EVENT_Q))  then
      begin
      DO_EVENT_66(
        GAME_TIME => LV_GAME_TIME,
        EVENT_Q => LV_EVENT_Q,
        SIMULATION_HISTORY => LV_SIMULATION_HISTORY);
      exception
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("DO_EVENT_66_65");
          EXCEPTION_HAS_OCCURRED := true;
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
      end;
    else return;
    end if;

 -- Exception Constraint translations.

 -- Other constraint option translations.

 -- Unconditional output translations.
    if not EXCEPTION_HAS_OCCURRED then
      begin
        DS_GAME_TIME_CREATE_NEW_EVENTS_114_113.BUFFER.WRITE(LV_GAME_TIME);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("GAME_TIME_CREATE_NEW_EVENTS_114_113",
                                                        "DO_EVENT_66_65");
      end;
      begin
        DS_GAME_TIME_DO_EVENT_66_65.BUFFER.WRITE(LV_GAME_TIME);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("GAME_TIME_DO_EVENT_66_65",
                                                    "DO_EVENT_66_65");
      end;
      begin
        DS_GAME_TIME_CREATE_USER_EVENT_69_68.BUFFER.WRITE(LV_GAME_TIME);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("GAME_TIME_CREATE_USER_EVENT_69_68",
                                                        "DO_EVENT_66_65");
      end;
    end if;
    if not EXCEPTION_HAS_OCCURRED then
      begin
        DS_EVENT_Q_CREATE_USER_EVENT_69_68.BUFFER.WRITE(LV_EVENT_Q);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("EVENT_Q_CREATE_USER_EVENT_69_68",
                                                        "DO_EVENT_66_65");
      end;
      begin
        DS_EVENT_Q_CREATE_NEW_EVENTS_114_113.BUFFER.WRITE(LV_EVENT_Q);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("EVENT_Q_CREATE_NEW_EVENTS_114_113",
                                                        "DO_EVENT_66_65");
```

```
    end;
    begin
      DS_EVENT_Q_DO_EVENT_66_65.BUFFER.WRITE(LV_EVENT_Q);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("EVENT_Q_DO_EVENT_66_65",
                                                  "DO_EVENT_66_65");
    end;
  end if;
  if not EXCEPTION_HAS_OCCURRED then
    begin
      DS_SIMULATION_HISTORY_POST_PROCESSOR_6_5.BUFFER.WRITE(
                                              LV_SIMULATION_HISTORY);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("SIMULATION_HISTORY_POST_PROCESSOR_6_5",
                                                  "DO_EVENT_66_65");
    end;
    begin
      DS_SIMULATION_HISTORY_JAAWS_12_11.BUFFER.WRITE(
                                              LV_SIMULATION_HISTORY);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("SIMULATION_HISTORY_JAAWS_12_11",
                                                  "DO_EVENT_66_65");
    end;
    begin
      DS_SIMULATION_HISTORY_DO_EVENT_66_65.BUFFER.WRITE(
                                              LV_SIMULATION_HISTORY);
    exception
      when BUFFER_OVERFLOW =>
        DS_DEBUG.BUFFER_OVERFLOW("SIMULATION_HISTORY_DO_EVENT_66_65",
                                                  "DO_EVENT_66_65");
    end;
  end if;

-- PSDL Exception handler.
  if EXCEPTION_HAS_OCCURRED then
    DS_DEBUG.UNHANDLED_EXCEPTION(
      "DO_EVENT_66_65",
      PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
  end if;
end DO_EVENT_66_65_DRIVER;


procedure CREATE_USER_EVENT_69_68_DRIVER is
  LV_GAME_TIME : GAME_TIME_TYPE_PKG.GAME_TIME_TYPE;
  LV_USER_INTERACTION : USER_INTERACTION_TYPE_PKG.USER_INTERACTION_TYPE;
  LV_EVENT_Q : EVENT_QUEUE_TYPE_PKG.EVENT_QUEUE_TYPE;

  EXCEPTION_HAS_OCCURRED: BOOLEAN := FALSE;
  EXCEPTION_ID: PSDL_EXCEPTION;
begin
-- Data trigger checks.

-- Data stream reads.
  begin
    DS_GAME_TIME_CREATE_USER_EVENT_69_68.BUFFER.READ(LV_GAME_TIME);
  exception
    when BUFFER_UNDERFLOW =>
      DS_DEBUG.BUFFER_UNDERFLOW("GAME_TIME_CREATE_USER_EVENT_69_68",
                                        "CREATE_USER_EVENT_69_68");
  end;
```

```
    begin
      DS_EVENT_Q_CREATE_USER_EVENT_69_68.BUFFER.READ(LV_EVENT_Q);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("EVENT_Q_CREATE_USER_EVENT_69_68",
                                  "CREATE_USER_EVENT_69_68");
    end;
    begin
      DS_USER_INTERACTION_CREATE_USER_EVENT_69_68.BUFFER.READ(
                                          LV_USER_INTERACTION);
    exception
      when BUFFER_UNDERFLOW =>
        DS_DEBUG.BUFFER_UNDERFLOW("USER_INTERACTION_CREATE_USER_EVENT_69_68",
                                  "CREATE_USER_EVENT_69_68");
    end;

 -- Execution trigger condition check.
    if  (LV_USER_INTERACTION = STOP_SIMULATION)  then
      begin
      CREATE_USER_EVENT_69(
        GAME_TIME => LV_GAME_TIME,
        USER_INTERACTION => LV_USER_INTERACTION,
        EVENT_Q => LV_EVENT_Q);
      exception
        when others =>
          DS_DEBUG.UNDECLARED_EXCEPTION("CREATE_USER_EVENT_69_68");
          EXCEPTION_HAS_OCCURRED := true;
          EXCEPTION_ID := UNDECLARED_ADA_EXCEPTION;
      end;
    else return;
    end if;

 -- Exception Constraint translations.

 -- Other constraint option translations.

 -- Unconditional output translations.
    if not EXCEPTION_HAS_OCCURRED then
      begin
        DS_EVENT_Q_CREATE_USER_EVENT_69_68.BUFFER.WRITE(LV_EVENT_Q);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("EVENT_Q_CREATE_USER_EVENT_69_68",
                                  "CREATE_USER_EVENT_69_68");
      end;
      begin
        DS_EVENT_Q_CREATE_NEW_EVENTS_114_113.BUFFER.WRITE(LV_EVENT_Q);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("EVENT_Q_CREATE_NEW_EVENTS_114_113",
                                  "CREATE_USER_EVENT_69_68");
      end;
      begin
        DS_EVENT_Q_DO_EVENT_66_65.BUFFER.WRITE(LV_EVENT_Q);
      exception
        when BUFFER_OVERFLOW =>
          DS_DEBUG.BUFFER_OVERFLOW("EVENT_Q_DO_EVENT_66_65",
                                  "CREATE_USER_EVENT_69_68");
      end;
    end if;

 -- PSDL Exception handler.
    if EXCEPTION_HAS_OCCURRED then
```

```
      DS_DEBUG.UNHANDLED_EXCEPTION(
        "CREATE_USER_EVENT_69_68",
        PSDL_EXCEPTION'IMAGE(EXCEPTION_ID));
    end if;
  end CREATE_USER_EVENT_69_68_DRIVER;
end WARRIOR_1_DRIVERS;
```

4. WARRIOR_1_EXCEPTIONS.ADS

```
package WARRIOR_1_EXCEPTIONS is
  -- PSDL exception type declaration
  type PSDL_EXCEPTION is (UNDECLARED_ADA_EXCEPTION);
end WARRIOR_1_EXCEPTIONS;
```

5. WARRIOR_1_INSTANTIATIONS.ADS

```
  with EVENT_TYPE_PKG; use EVENT_TYPE_PKG;
  -- Generic type packages
  with
    SEQUENCE_PKG;
package WARRIOR_1_INSTANTIATIONS is
  -- Ada Generic package instantiations

  package EVENT_TYPE_SEQUENCE_PKG is new
    SEQUENCE_PKG(EVENT_TYPE_PTR);

  type EVENT_TYPE_SEQUENCE is new
    EVENT_TYPE_SEQUENCE_PKG.SEQUENCE;

end WARRIOR_1_INSTANTIATIONS;
```

6. WARRIOR_1_STREAMS.ADS

```
  -- with/use clauses for atomic type packages
  with EVENT_TYPE_PKG; use EVENT_TYPE_PKG;
  with EVENT_QUEUE_TYPE_PKG; use EVENT_QUEUE_TYPE_PKG;
  with STATISTICS_TYPE_PKG; use STATISTICS_TYPE_PKG;
  with SCENARIO_TYPE_PKG; use SCENARIO_TYPE_PKG;
  with STATISTICS_REQUEST_TYPE_PKG; use STATISTICS_REQUEST_TYPE_PKG;
  with REPLAY_REQUEST_TYPE_PKG; use REPLAY_REQUEST_TYPE_PKG;
  with USER_INTERACTION_TYPE_PKG; use USER_INTERACTION_TYPE_PKG;
  with LOCATION_TYPE_PKG; use LOCATION_TYPE_PKG;
  with GAME_TIME_TYPE_PKG; use GAME_TIME_TYPE_PKG;
  -- with/use clauses for generated packages.
  with WARRIOR_1_EXCEPTIONS; use WARRIOR_1_EXCEPTIONS;
  with WARRIOR_1_INSTANTIATIONS; use WARRIOR_1_INSTANTIATIONS;
  -- with/use clauses for CAPS library packages.
  with PSDL_STREAMS; use PSDL_STREAMS;
  with PSDL_STRING_PKG; use PSDL_STRING_PKG;
package WARRIOR_1_STREAMS is
  -- Local stream instantiations

  package DS_USER_INTERACTION_CREATE_USER_EVENT_69_68 is new
    PSDL_STREAMS.SAMPLED_BUFFER(USER_INTERACTION_TYPE);

  package DS_STATISTICS_REQUEST_POST_PROCESSOR_6_5 is new
    PSDL_STREAMS.FIFO_BUFFER(STATISTICS_REQUEST_TYPE);

  package DS_STATISTICS_DISPLAY_ST_31_30 is new
    PSDL_STREAMS.SAMPLED_BUFFER(STATISTICS_TYPE);
```

```
package DS_REPLAY_DISPLAY_RE_37_36 is new
  PSDL_STREAMS.SAMPLED_BUFFER(LOCATION_TYPE);

package DS_SIMULATION_HISTORY_POST_PROCESSOR_6_5 is new
  PSDL_STREAMS.SAMPLED_BUFFER(EVENT_TYPE_SEQUENCE);

package DS_SIMULATION_HISTORY_JAAWS_12_11 is new
  PSDL_STREAMS.SAMPLED_BUFFER(EVENT_TYPE_SEQUENCE);

package DS_SIMULATION_HISTORY_DO_EVENT_66_65 is new
  PSDL_STREAMS.SAMPLED_BUFFER(EVENT_TYPE_SEQUENCE);

package DS_NEW_PLAN_ENTERED_EDIT_PLAN_24_23 is new
  PSDL_STREAMS.FIFO_BUFFER(BOOLEAN);

-- State stream instantiations

package DS_REPLAY_POSITION_JAAWS_12_11 is new
        PSDL_STREAMS.STATE_VARIABLE(INTEGER, 1);

package DS_REPLAY_REQUEST_JAAWS_12_11 is new
        PSDL_STREAMS.STATE_VARIABLE(
              REPLAY_REQUEST_TYPE_PKG.REPLAY_REQUEST_TYPE,
                                     REPLAY_REQUEST_TYPE_PKG.OFF);

package DS_SCENARIO_CREATE_NEW_EVENTS_114_113 is new
        PSDL_STREAMS.STATE_VARIABLE(
              SCENARIO_TYPE_PKG.SCENARIO_TYPE,
                                SCENARIO_TYPE_PKG.EMPTY_SCENARIO);

package DS_SCENARIO_EDIT_PLAN_24_23 is new
        PSDL_STREAMS.STATE_VARIABLE(
              SCENARIO_TYPE_PKG.SCENARIO_TYPE,
                          SCENARIO_TYPE_PKG.EMPTY_SCENARIO);

package DS_NEW_Y_EDIT_PLAN_24_23 is new
        PSDL_STREAMS.STATE_VARIABLE(FLOAT, 0.0);

package DS_NEW_X_EDIT_PLAN_24_23 is new
        PSDL_STREAMS.STATE_VARIABLE(FLOAT, 0.0);

package DS_FIRST_TIME_INITIAL_SCENARIO_40_39 is new
        PSDL_STREAMS.STATE_VARIABLE(BOOLEAN, true);

package DS_GAME_TIME_CREATE_NEW_EVENTS_114_113 is new
        PSDL_STREAMS.STATE_VARIABLE(
                    GAME_TIME_TYPE_PKG.GAME_TIME_TYPE,
                                  GAME_TIME_TYPE_PKG.ZERO);

package DS_GAME_TIME_DO_EVENT_66_65 is new
        PSDL_STREAMS.STATE_VARIABLE(
                    GAME_TIME_TYPE_PKG.GAME_TIME_TYPE,
                                  GAME_TIME_TYPE_PKG.ZERO);

package DS_GAME_TIME_CREATE_USER_EVENT_69_68 is new
        PSDL_STREAMS.STATE_VARIABLE(
```

```
                         GAME_TIME_TYPE_PKG.GAME_TIME_TYPE,
                                     GAME_TIME_TYPE_PKG.ZERO);

    package DS_EVENT_Q_CREATE_USER_EVENT_69_68 is new
            PSDL_STREAMS.STATE_VARIABLE(
                         EVENT_QUEUE_TYPE_PKG.EVENT_QUEUE_TYPE,
                                     EVENT_QUEUE_TYPE_PKG.EMPTY);

    package DS_EVENT_Q_CREATE_NEW_EVENTS_114_113 is new
            PSDL_STREAMS.STATE_VARIABLE(
                         EVENT_QUEUE_TYPE_PKG.EVENT_QUEUE_TYPE,
                                     EVENT_QUEUE_TYPE_PKG.EMPTY);

    package DS_EVENT_Q_DO_EVENT_66_65 is new
            PSDL_STREAMS.STATE_VARIABLE(
                         EVENT_QUEUE_TYPE_PKG.EVENT_QUEUE_TYPE,
                                     EVENT_QUEUE_TYPE_PKG.EMPTY);

  end WARRIOR_1_STREAMS;
```

7. WARRIOR_1_TIMERS.ADS

```
  with PSDL_TIMERS;
package WARRIOR_1_TIMERS is
  -- Timer instantiations
end WARRIOR_1_TIMERS;
```

8. WARRIOR_1_DYNAMIC_SCHEDULERS.ADS

```
package warrior_1_DYNAMIC_SCHEDULERS is
  procedure START_DYNAMIC_SCHEDULE;
  procedure STOP_DYNAMIC_SCHEDULE;
end warrior_1_DYNAMIC_SCHEDULERS;
```

9. WARRIOR_1_DYNAMIC_SCHEDULERS.ADB

```
with warrior_1_DRIVERS;  use warrior_1_DRIVERS;
with PRIORITY_DEFINITIONS;  use PRIORITY_DEFINITIONS;
package body warrior_1_DYNAMIC_SCHEDULERS is

  task type DYNAMIC_SCHEDULE_TYPE is
    pragma priority (DYNAMIC_SCHEDULE_PRIORITY);
    entry START;
  end DYNAMIC_SCHEDULE_TYPE;
  for DYNAMIC_SCHEDULE_TYPE'STORAGE_SIZE use 100_000;
  DYNAMIC_SCHEDULE : DYNAMIC_SCHEDULE_TYPE;

  done : boolean := false;
  procedure STOP_DYNAMIC_SCHEDULE is
  begin
    done := true;
  end STOP_DYNAMIC_SCHEDULE;

  task body DYNAMIC_SCHEDULE_TYPE is
  begin
    accept START;
    loop
      enter_new_plan_75_74_DRIVER;
      exit when done;

      get_y_68_67_DRIVER;
```

```
        exit when done;

        get_x_65_64_DRIVER;
        exit when done;

        get_re_30_29_DRIVER;
        exit when done;

        get_st_27_26_DRIVER;
        exit when done;

        get_user_in_21_20_DRIVER;
        exit when done;

        initial_scenario_40_39_DRIVER;
        exit when done;

        create_new_events_114_113_DRIVER;
        exit when done;

        edit_plan_24_23_DRIVER;
        exit when done;

        create_user_event_69_68_DRIVER;
        exit when done;

        jaaws_12_11_DRIVER;
        exit when done;

        post_processor_6_5_DRIVER;
        exit when done;

        display_re_37_36_DRIVER;
        exit when done;

        display_st_31_30_DRIVER;
        exit when done;

    end loop;
  end DYNAMIC_SCHEDULE_TYPE;

  procedure START_DYNAMIC_SCHEDULE is
  begin
    DYNAMIC_SCHEDULE.START;
  end START_DYNAMIC_SCHEDULE;

end warrior_1_DYNAMIC_SCHEDULERS;
```

10. WARRIOR_1_STATIC_SCHEDULERS.ADS

```
package warrior_1_STATIC_SCHEDULERS is
  procedure START_STATIC_SCHEDULE;
  procedure STOP_STATIC_SCHEDULE;
end warrior_1_STATIC_SCHEDULERS;
```

11. WARRIOR_1_STATIC_SCHEDULERS.ADB

```
with warrior_1_DRIVERS;  use warrior_1_DRIVERS;
with PRIORITY_DEFINITIONS;  use PRIORITY_DEFINITIONS;
with PSDL_TIMERS;  use PSDL_TIMERS;
with TEXT_IO;  use TEXT_IO;
package body warrior_1_STATIC_SCHEDULERS is

  task type STATIC_SCHEDULE_TYPE is
    pragma priority (STATIC_SCHEDULE_PRIORITY);
    entry START;
  end STATIC_SCHEDULE_TYPE;
  for STATIC_SCHEDULE_TYPE'STORAGE_SIZE use 200_000;
  STATIC_SCHEDULE : STATIC_SCHEDULE_TYPE;

  done : boolean := false;
  procedure STOP_STATIC_SCHEDULE is
  begin
    done := true;
  end STOP_STATIC_SCHEDULE;

  task body STATIC_SCHEDULE_TYPE is
    PERIOD : duration;
    gui_event_monitor_18_17_START_TIME1 : duration;
    gui_event_monitor_18_17_STOP_TIME1 : duration;
    do_event_66_65_START_TIME2 : duration;
    do_event_66_65_STOP_TIME2 : duration;
    gui_event_monitor_18_17_START_TIME3 : duration;
    gui_event_monitor_18_17_STOP_TIME3 : duration;
    gui_event_monitor_18_17_START_TIME4 : duration;
    gui_event_monitor_18_17_STOP_TIME4 : duration;
    gui_event_monitor_18_17_START_TIME5 : duration;
    gui_event_monitor_18_17_STOP_TIME5 : duration;
    do_event_66_65_START_TIME6 : duration;
    do_event_66_65_STOP_TIME6 : duration;
    gui_event_monitor_18_17_START_TIME7 : duration;
    gui_event_monitor_18_17_STOP_TIME7 : duration;
    gui_event_monitor_18_17_START_TIME8 : duration;
    gui_event_monitor_18_17_STOP_TIME8 : duration;
    gui_event_monitor_18_17_START_TIME9 : duration;
    gui_event_monitor_18_17_STOP_TIME9 : duration;
    do_event_66_65_START_TIME10 : duration;
    do_event_66_65_STOP_TIME10 : duration;
    gui_event_monitor_18_17_START_TIME11 : duration;
    gui_event_monitor_18_17_STOP_TIME11 : duration;
    gui_event_monitor_18_17_START_TIME12 : duration;
    gui_event_monitor_18_17_STOP_TIME12 : duration;
    gui_event_monitor_18_17_START_TIME13 : duration;
    gui_event_monitor_18_17_STOP_TIME13 : duration;
    schedule_timer : TIMER := NEW_TIMER;
  begin
    accept START;
    PERIOD := TARGET_TO_HOST(duration( 3.00000E+00));
    gui_event_monitor_18_17_START_TIME1 := TARGET_TO_HOST(
                                          duration( 0.00000E+00));
    gui_event_monitor_18_17_STOP_TIME1 := TARGET_TO_HOST(
                                          duration( 5.00000E-02));
    do_event_66_65_START_TIME2 := TARGET_TO_HOST(duration( 5.00000E-02));
    do_event_66_65_STOP_TIME2 := TARGET_TO_HOST(duration( 1.50000E-01));
    gui_event_monitor_18_17_START_TIME3 := TARGET_TO_HOST(
                                          duration( 3.00000E-01));
    gui_event_monitor_18_17_STOP_TIME3 := TARGET_TO_HOST(
                                          duration( 3.50000E-01));
```

```
gui_event_monitor_18_17_START_TIME4 := TARGET_TO_HOST(
                                        duration( 6.00000E-01));
gui_event_monitor_18_17_STOP_TIME4 := TARGET_TO_HOST(
                                        duration( 6.50000E-01));
gui_event_monitor_18_17_START_TIME5 := TARGET_TO_HOST(
                                        duration( 9.00000E-01));
gui_event_monitor_18_17_STOP_TIME5 := TARGET_TO_HOST(
                                        duration( 9.50000E-01));
do_event_66_65_START_TIME6 := TARGET_TO_HOST(duration( 1.05000E+00));
do_event_66_65_STOP_TIME6 := TARGET_TO_HOST(duration( 1.15000E+00));
gui_event_monitor_18_17_START_TIME7 := TARGET_TO_HOST(
                                        duration( 1.20000E+00));
gui_event_monitor_18_17_STOP_TIME7 := TARGET_TO_HOST(
                                        duration( 1.25000E+00));
gui_event_monitor_18_17_START_TIME8 := TARGET_TO_HOST(
                                        duration( 1.50000E+00));
gui_event_monitor_18_17_STOP_TIME8 := TARGET_TO_HOST(
                                        duration( 1.55000E+00));
gui_event_monitor_18_17_START_TIME9 := TARGET_TO_HOST(
                                        duration( 1.80000E+00));
gui_event_monitor_18_17_STOP_TIME9 := TARGET_TO_HOST(
                                        duration( 1.85000E+00));
do_event_66_65_START_TIME10 := TARGET_TO_HOST(duration( 2.05000E+00));
do_event_66_65_STOP_TIME10 := TARGET_TO_HOST(duration( 2.15000E+00));
gui_event_monitor_18_17_START_TIME11 := TARGET_TO_HOST(
                                        duration( 2.15000E+00));
gui_event_monitor_18_17_STOP_TIME11 := TARGET_TO_HOST(
                                        duration( 2.20000E+00));
gui_event_monitor_18_17_START_TIME12 := TARGET_TO_HOST(
                                        duration( 2.40000E+00));
gui_event_monitor_18_17_STOP_TIME12 := TARGET_TO_HOST(
                                        duration( 2.45000E+00));
gui_event_monitor_18_17_START_TIME13 := TARGET_TO_HOST(
                                        duration( 2.70000E+00));
gui_event_monitor_18_17_STOP_TIME13 := TARGET_TO_HOST(
                                        duration( 2.75000E+00));
START(schedule_timer);
loop
  delay(gui_event_monitor_18_17_START_TIME1 -
                            HOST_DURATION(schedule_timer));
  gui_event_monitor_18_17_DRIVER;
  if HOST_DURATION(schedule_timer) >
                          gui_event_monitor_18_17_STOP_TIME1 then
    PUT_LINE("timing error from operator gui_event_monitor_18_17");
    SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
                                 gui_event_monitor_18_17_STOP_TIME1);
  end if;
  exit when done;

  delay(do_event_66_65_START_TIME2 - HOST_DURATION(schedule_timer));
  do_event_66_65_DRIVER;
  if HOST_DURATION(schedule_timer) > do_event_66_65_STOP_TIME2 then
    PUT_LINE("timing error from operator do_event_66_65");
    SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
                                        do_event_66_65_STOP_TIME2);
  end if;
  exit when done;

  delay(gui_event_monitor_18_17_START_TIME3 -
                                 HOST_DURATION(schedule_timer));
  gui_event_monitor_18_17_DRIVER;
  if HOST_DURATION(schedule_timer) >
                                 gui_event_monitor_18_17_STOP_TIME3 then
```

```
  PUT_LINE("timing error from operator gui_event_monitor_18_17");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
                              gui_event_monitor_18_17_STOP_TIME3);
end if;
exit when done;

delay(gui_event_monitor_18_17_START_TIME4 -
                                     HOST_DURATION(schedule_timer));
gui_event_monitor_18_17_DRIVER;
if HOST_DURATION(schedule_timer) >
                              gui_event_monitor_18_17_STOP_TIME4 then
  PUT_LINE("timing error from operator gui_event_monitor_18_17");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
                              gui_event_monitor_18_17_STOP_TIME4);
end if;
exit when done;

delay(gui_event_monitor_18_17_START_TIME5 -
                              HOST_DURATION(schedule_timer));
gui_event_monitor_18_17_DRIVER;
if HOST_DURATION(schedule_timer) >
                              gui_event_monitor_18_17_STOP_TIME5 then
  PUT_LINE("timing error from operator gui_event_monitor_18_17");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
                              gui_event_monitor_18_17_STOP_TIME5);
end if;
exit when done;

delay(do_event_66_65_START_TIME6 - HOST_DURATION(schedule_timer));
do_event_66_65_DRIVER;
if HOST_DURATION(schedule_timer) > do_event_66_65_STOP_TIME6 then
  PUT_LINE("timing error from operator do_event_66_65");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
                                     do_event_66_65_STOP_TIME6);
end if;
exit when done;

delay(gui_event_monitor_18_17_START_TIME7 -
                                     HOST_DURATION(schedule_timer));
gui_event_monitor_18_17_DRIVER;
if HOST_DURATION(schedule_timer) >
                              gui_event_monitor_18_17_STOP_TIME7 then
  PUT_LINE("timing error from operator gui_event_monitor_18_17");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
                              gui_event_monitor_18_17_STOP_TIME7);
end if;
exit when done;

delay(gui_event_monitor_18_17_START_TIME8 -
                                     HOST_DURATION(schedule_timer));
gui_event_monitor_18_17_DRIVER;
if HOST_DURATION(schedule_timer) >
                              gui_event_monitor_18_17_STOP_TIME8 then
  PUT_LINE("timing error from operator gui_event_monitor_18_17");
  SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
                              gui_event_monitor_18_17_STOP_TIME8);
end if;
exit when done;

delay(gui_event_monitor_18_17_START_TIME9 -
                                     HOST_DURATION(schedule_timer));
gui_event_monitor_18_17_DRIVER;
if HOST_DURATION(schedule_timer) >
```

```
                                    gui_event_monitor_18_17_STOP_TIME9 then
      PUT_LINE("timing error from operator gui_event_monitor_18_17");
      SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
                                    gui_event_monitor_18_17_STOP_TIME9);
    end if;
    exit when done;

    delay(do_event_66_65_START_TIME10 - HOST_DURATION(schedule_timer));
    do_event_66_65_DRIVER;
    if HOST_DURATION(schedule_timer) > do_event_66_65_STOP_TIME10 then
      PUT_LINE("timing error from operator do_event_66_65");
      SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
                                        do_event_66_65_STOP_TIME10);
    end if;
    exit when done;

    delay(gui_event_monitor_18_17_START_TIME11 -
                                        HOST_DURATION(schedule_timer));
    gui_event_monitor_18_17_DRIVER;
    if HOST_DURATION(schedule_timer) >
                                gui_event_monitor_18_17_STOP_TIME11 then
      PUT_LINE("timing error from operator gui_event_monitor_18_17");
      SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
                                    gui_event_monitor_18_17_STOP_TIME11);
    end if;
    exit when done;

    delay(gui_event_monitor_18_17_START_TIME12 -
                                        HOST_DURATION(schedule_timer));
    gui_event_monitor_18_17_DRIVER;
    if HOST_DURATION(schedule_timer) >
                                gui_event_monitor_18_17_STOP_TIME12 then
      PUT_LINE("timing error from operator gui_event_monitor_18_17");
      SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
                                    gui_event_monitor_18_17_STOP_TIME12);
    end if;
    exit when done;

    delay(gui_event_monitor_18_17_START_TIME13 -
                                        HOST_DURATION(schedule_timer));
    gui_event_monitor_18_17_DRIVER;
    if HOST_DURATION(schedule_timer) >
                                gui_event_monitor_18_17_STOP_TIME13 then
      PUT_LINE("timing error from operator gui_event_monitor_18_17");
      SUBTRACT_HOST_TIME_FROM_ALL_TIMERS(HOST_DURATION(schedule_timer) -
                                    gui_event_monitor_18_17_STOP_TIME13);
    end if;
    exit when done;

    delay(PERIOD - HOST_DURATION(schedule_timer));
    RESET(schedule_timer);
  end loop;
end STATIC_SCHEDULE_TYPE;

procedure START_STATIC_SCHEDULE is
begin
  STATIC_SCHEDULE.START;
end START_STATIC_SCHEDULE;

end warrior_1_STATIC_SCHEDULERS;
```

12. WARRIOR_EVENT_MONITOR_TASK_PKG.ADS

```
  -- The wrapper task to provide mutual exclusion
  -- for calls from the prototype to TAE.

  with PRIORITY_DEFINITIONS; use PRIORITY_DEFINITIONS;
  with statistics_type_pkg; use statistics_type_pkg;
  with location_type_pkg; use location_type_pkg;
package warrior_event_monitor_task_pkg is
  task warrior_event_monitor_task is
    pragma priority (BUFFER_PRIORITY);
    entry event_monitor_entry;
    entry display_st_31_entry(statistics: statistics_type);
    entry display_re_37_entry(replay: location_type);
    entry end_task;
  end warrior_event_monitor_task;
end warrior_event_monitor_task_pkg;
```

13. WARRIOR_EVENT_MONITOR_TASK_PKG.ADB

```
  -- The wrapper task to provide mutual exclusion
  -- for calls from the prototype to TAE.

  with generated_tae_event_monitor_pkg;
  with panel_gui_3;
with text_io;
package body warrior_event_monitor_task_pkg is
  task body warrior_event_monitor_task is
    done : boolean := false;
  begin
    panel_gui_3.initialize_gui;
    loop
      select
        accept event_monitor_entry do
         if not done then
            generated_tae_event_monitor_pkg.generated_tae_event_monitor;
         end if;
        end event_monitor_entry;
      or
        accept display_st_31_entry(statistics: statistics_type) do
         if not done then
            panel_gui_3.display_st_31(statistics);
         end if;
        end display_st_31_entry;
      or
        accept display_re_37_entry(replay: location_type) do
         if not done then
            panel_gui_3.display_re_37(replay);
         end if;
        end display_re_37_entry;
      or
        accept end_task do
         raise Program_Error;
        end end_task;
      end select;
    end loop;

  end warrior_event_monitor_task;
end warrior_event_monitor_task_pkg;
```

14. CREATE_NEW_EVENTS_114_PKG.ADS

```
  with game_time_type_pkg;   use game_time_type_pkg;
  with event_queue_type_pkg; use event_queue_type_pkg;
  with scenario_type_pkg;    use scenario_type_pkg;

package create_new_events_114_pkg is

  procedure create_new_events_114( game_time: in     game_time_type;
                                     event_q: in out event_queue_type;
                                    scenario: in     scenario_type );
end create_new_events_114_pkg;
```

15. CREATE_NEW_EVENTS_114_PKG.ADB

```
  with simulation_object_Pkg;       USE Simulation_Object_Pkg;
  with event_type_pkg;              USE event_type_pkg;
  with event_type_pkg.move_pkg;     use event_type_pkg.move_pkg;
  with text_io;

package body create_new_events_114_pkg is
  procedure create_new_events_114( game_time: in     game_time_type;
                                     event_q: in out event_queue_type;
                                    scenario: in     scenario_type ) is

      Event      : Event_Type_Ptr;
      Object_Ptr : Simulation_Object_Ptr;

  begin --
    Object_Ptr := Get_Unit ( Scenario ); -- Just one unit in this version
    if Can_move(Object_Ptr.all) and      -- Just one kind of initial event
       not Get_Is_Scheduled(Object_Ptr.all)
    then
       -- since this is currently the only type of event, move
       Event := event_type_pkg.move_pkg.Construct_Event (Object_Ptr,
                                                               Game_Time );
       Schedule_Event( Event, Event_Q );
       Set_Is_Scheduled(Object_Ptr.all, true);
    end if;
  end create_new_events_114;
end create_new_events_114_pkg;
```

16. CREATE_USER_EVENT_69_PKG.ADS

```
  with game_time_type_pkg;        use game_time_type_pkg;
  with event_queue_type_pkg;      use event_queue_type_pkg;
  with user_interaction_type_pkg; use user_interaction_type_pkg;

package create_user_event_69_pkg is

  procedure create_user_event_69( game_time: in     game_time_type;
                                    event_q: in out event_queue_type;
                           user_interaction: in     user_interaction_type );
end create_user_event_69_pkg;
```

17. CREATE_USER_EVENT_69_PKG.ADB

```
  WITH Simulation_Object_Pkg;  USE Simulation_Object_Pkg;
  WITH Event_Type_Pkg;         USE Event_Type_Pkg;
  with event_type_pkg.end_sim_pkg; use event_type_pkg.end_sim_pkg;

package body create_user_event_69_pkg is
```

```
  procedure create_user_event_69( game_time: in     game_time_type;
                                    event_q: in out event_queue_type;
                           user_interaction: in     user_interaction_type ) is

      Event      : Event_Type_Ptr;
      Object_Ptr : Simulation_Object_Ptr := NULL;

  begin --
    if User_Interaction = stop_simulation then
       -- Only one kind of user interaction in this version.
       Event := event_type_pkg.end_sim_pkg.Construct_Event( Object_Ptr,
                                                             Game_Time );

       Schedule_Event( Event, event_q );
    end if;

  end create_user_event_69;
end create_user_event_69_pkg;
```

18. DELIMITER_PKG.ADS

```
package delimiter_pkg is
  type delimiter_array is array (character) of boolean;
  function initialize_delimiter_array return delimiter_array;
end delimiter_pkg;
```

19. DELIMITER_PKG.ADB

```
package body delimiter_pkg is
  function initialize_delimiter_array return delimiter_array is
  begin
    return (' ' | ascii.ht | ascii.cr | ascii.lf => true, others => false);
  end initialize_delimiter_array;
end delimiter_pkg;
```

20. DISPLAY_RE_37_PKG.ADS

```
with location_type_pkg; use location_type_pkg;
package display_re_37_pkg is
  procedure display_re_37(replay: location_type);
end display_re_37_pkg;
```

21. DISPLAY_RE_37_PKG.ADB

```
  with warrior_event_monitor_task_pkg;
  use warrior_event_monitor_task_pkg;
package body display_re_37_pkg is
  procedure display_re_37(replay: location_type) is
  begin
    warrior_event_monitor_task.display_re_37_entry(replay);
  end display_re_37;
end display_re_37_pkg;
```

22. DISPLAY_ST_31_PKG.ADS

```
with statistics_type_pkg; use statistics_type_pkg;
package display_st_31_pkg is
  procedure display_st_31(statistics: statistics_type);
end display_st_31_pkg;
```

23. DISPLAY_ST_31_PKG.ADB

```
  with warrior_event_monitor_task_pkg;
  use warrior_event_monitor_task_pkg;
package body display_st_31_pkg is
  procedure display_st_31(statistics: statistics_type) is
  begin
    warrior_event_monitor_task.display_st_31_entry(statistics);
  end display_st_31;
end display_st_31_pkg;
```

24. DO_EVENT_66_PKG.ADS

```
  with game_time_type_pkg;        use game_time_type_pkg;
  with event_queue_type_pkg;      use event_queue_type_pkg;
  with warrior_1_instantiations;  use warrior_1_instantiations;
  with warrior_1_exceptions;      use warrior_1_exceptions;

package do_event_66_pkg is

  procedure do_event_66( game_time: in out game_time_type;
                 simulation_history: in out event_type_sequence;
                            event_q: in out event_queue_type );
end do_event_66_pkg;
```

25. DO_EVENT_66_PKG.ADB

```
  with simulation_object_pkg;        use simulation_object_pkg;
  with event_type_pkg;               use event_type_pkg;
  with event_type_pkg.move_pkg;      use event_type_pkg.move_pkg;
  with event_type_pkg.end_sim_pkg;   use event_type_pkg.end_sim_pkg;


package body do_event_66_pkg is

  procedure do_event_66( game_time: in out game_time_type;
                         simulation_history: in out event_type_sequence;
                         event_q: in out event_queue_type ) is

     Next_Time    : Game_Time_Type;
     Event        : Event_Type_Ptr;
  begin --
    Get_Next_Event( Event, event_q );     -- get event from event queue
    Next_Time := Execute_Event( Event.ALL );  -- execute event and get next
                                              -- execution time
    Game_Time := Get_Event_Time( Event.ALL ); -- update game time to time of
                                              -- event
    Simulation_History := Add( Copy_Event( Event.ALL ), Simulation_History );
    if Next_Time /= NEVER then
       Set_Event_Time( Event.ALL, Next_Time );
       Schedule_Event( Event, event_q );
    end if;

  end do_event_66;
end do_event_66_pkg;
```

26. EDIT_PLAN_24_PKG.ADS

```
with scenario_type_pkg;         use scenario_type_pkg;
  with warrior_1_instantiations; use warrior_1_instantiations;
  with warrior_1_exceptions;     use warrior_1_exceptions;

package edit_plan_24_pkg is
```

```
  procedure edit_plan_24( new_plan_entered: in boolean;
                          new_y: in     float;
                          new_x: in     float;
                       scenario: in out scenario_type );
end edit_plan_24_pkg;
```

27. EDIT_PLAN_24_PKG.ADB

```
with Location_Type_pkg;     use Location_Type_pkg;
  with Simulation_Object_Pkg; use Simulation_Object_Pkg;

package body edit_plan_24_pkg is

  procedure edit_plan_24( new_plan_entered: in boolean;
                          new_y: in     float;
                          new_x: in     float;
                       scenario: in out scenario_type ) is

    unit: Simulation_Object_Ptr;
    destination: Location_Type;

  begin --
    destination := To_Location(new_x, new_y);
    unit := get_unit(scenario);
    Set_Destination(unit.all, destination);

  end edit_plan_24;
end edit_plan_24_pkg;
```

28. ENTER_NEW_PLAN_75_PKG.ADS

```
package enter_new_plan_75_pkg is
  procedure enter_new_plan_75(new_plan_entered : out boolean);
  procedure record_input(new_plan_entered : in boolean);
  function has_new_input return boolean;
    -- True iff a user input has arrived
    -- since the last time this bubble was executed.
end enter_new_plan_75_pkg;
```

29. ENTER_NEW_PLAN_75_PKG.ADB

```
with psdl_streams; use psdl_streams;
package body enter_new_plan_75_pkg is
  package new_plan_entered_buffer is new
    sampled_buffer(boolean);
  use new_plan_entered_buffer;

  procedure enter_new_plan_75(new_plan_entered : out boolean) is
  begin
    -- Get the value from new_plan_entered_buffer
    buffer.read(new_plan_entered);
  end enter_new_plan_75;

  procedure record_input(new_plan_entered : in boolean) is
  begin
    -- Save the value in new_plan_entered_buffer
    buffer.write(new_plan_entered);
  end record_input;

  function has_new_input return boolean is
  begin
    -- Check status of new_plan_entered_buffer
    return new_data;
```

```
  end has_new_input;

end enter_new_plan_75_pkg;
```

30. EVENT_QUEUE_TYPE_PKG.ADS

```
  WITH sorted_list_pkg;
  WITH Event_Type_Pkg; use Event_Type_Pkg;

package Event_Queue_Type_Pkg is

  type Event_Queue_Type is private;

  PROCEDURE Schedule_Event (Event   : IN     Event_Type_Ptr;
                            Event_Q : IN OUT Event_Queue_Type);

  PROCEDURE Get_Next_Event(Event   : out    Event_Type_Ptr;
                           Event_Q : IN OUT Event_Queue_Type);

  FUNCTION Is_Empty(Event_Q : IN Event_Queue_Type) RETURN BOOLEAN;

  FUNCTION Empty RETURN Event_Queue_Type;

private
  package e_q_pkg is new sorted_list_pkg(element_type => Event_Type_Ptr,
                                         "<" => "<");
  type Event_Queue_Type is new e_q_pkg.sorted_list;

end Event_Queue_Type_Pkg;
```

31. EVENT_QUEUE_TYPE_PKG.ADB

```
  with event_type_pkg.move_pkg;     use event_type_pkg.move_pkg;
  with event_type_pkg.end_sim_pkg;  use event_type_pkg.end_sim_pkg;
  with ada.text_io;

package body Event_Queue_Type_Pkg is

  PROCEDURE Schedule_Event
            (Event   : IN     Event_Type_Ptr;
             Event_Q : IN OUT Event_Queue_Type) is
  begin
    add(Event_Q, Event);
  end Schedule_Event;



  PROCEDURE Get_Next_Event
            (Event   : out    Event_Type_Ptr;
             Event_Q : IN OUT Event_Queue_Type) is
  begin
    get_smallest(Event_Q, Event);
  end Get_Next_Event;



  FUNCTION Is_Empty(Event_Q : IN Event_Queue_Type) RETURN BOOLEAN is
  begin
    return e_q_pkg.is_empty(e_q_pkg.sorted_list(Event_Q));
  end Is_Empty;
```

```
  FUNCTION Empty RETURN Event_Queue_Type is
  begin
    return Event_Queue_Type(e_q_pkg.empty);
  end Empty;
end Event_Queue_Type_Pkg;
```

32. EVENT_TYPE_PKG.ADS

```
-------------------------------------------------------------------------------
--| FileName:    Event_Type_Pkg.ads
--| Author:      Julian Williams
--| Date:        10 October 1998
--| Project:     Janus/Warrior Combat Simulation for CAPS
--| Compiler:    ObjectAda for Windows Ver. 7.1.1 (Professional)
--| Description: This package describes basic functions and procedures
--|              involving event types in the Warrior Combat Simulation
--|              model.
-------------------------------------------------------------------------------
WITH Simulation_Object_Pkg; USE Simulation_Object_Pkg;
WITH Game_Time_Type_Pkg;    USE Game_Time_Type_pkg;

PACKAGE Event_Type_Pkg IS

    TYPE Event_Action_Type IS ( MoveUpdateObj, EndSimulation );

    TYPE Event_Type IS ABSTRACT TAGGED PRIVATE;
    TYPE Event_Type_Ptr IS ACCESS ALL Event_Type'Class;

    ---------------------------------------------------------------------------
    --| FUNCTION Get_Event_Time
    --| Pre:     An unexecuted event exsist.
    --| Post:    Start time for the event is returned.
    ---------------------------------------------------------------------------
    FUNCTION Get_Event_Time (Event: IN Event_Type'Class)
                                                RETURN Game_Time_Type;

    ---------------------------------------------------------------------------
    --| PROCEDURE Set_Event_Time
    --| Pre:
    --| Post:
    ---------------------------------------------------------------------------
    PROCEDURE Set_Event_Time (Event: IN OUT Event_Type'Class;
                                       Time: IN Game_Time_Type);

    ---------------------------------------------------------------------------
    --| FUNCTION Get_Object
    --| Pre: An event exist.
    --| Post: The object designated within the event is returned.
    ---------------------------------------------------------------------------
    FUNCTION Get_Object (Event: IN Event_Type'Class)
                                         RETURN Simulation_Object_Ptr;

    ---------------------------------------------------------------------------
    --| FUNCTION Get_Action
    --| Pre: An event exist.
    --| Post: The action on the object in the event is returned.
    ---------------------------------------------------------------------------
```

```
    FUNCTION Get_Action (Event: IN Event_Type'Class)
                                           RETURN Event_Action_Type;

    --------------------------------------------------------------------
    --| FUNCTION "<"
    --| Pre: Two event types exist.
    --| Post: The least valued event is returned.
    --------------------------------------------------------------------
    FUNCTION "<" (Left, Right: IN Event_Type_Ptr) RETURN Boolean;


    --------------------------------------------------------------------
    --| FUNCTION Execute_Event
    --| Pre:     A move event has been extracted from the event queue
    --|          and needs to be executed.
    --| Post:    Move event is executed and time executed is returned.
    --------------------------------------------------------------------
    FUNCTION Execute_Event (Event: IN Event_Type)
                                                 ETURN Game_Time_Type;

    --------------------------------------------------------------------
    --| PROCEDURE Copy_Event
    --| Pre:     An move event exist.
    --| Post:    The move event is copied and a pointer to the copy is
    --|          returned.
    --------------------------------------------------------------------
    FUNCTION Copy_Event (Event: IN Event_Type)
                                                 RETURN Event_Type_Ptr;

PRIVATE
    TYPE Event_Type IS ABSTRACT TAGGED
      RECORD
        Action      : Event_Action_Type;              -- desired action
                                                      -- to be performed
        Object_Ptr  : Simulation_Object_Ptr := NULL;  -- pointer to
                                                      -- simulation
                                                      -- object
        Time        : Game_Time_Type;                 -- time to start
                                                      -- event action
    END RECORD;
END Event_Type_Pkg;
```

33. EVENT_TYPE_PKG.ADB

```
------------------------------------------------------------------------
--| FileName:    Event_Type_Pkg.adb
--| Author:      Julian Williams
--| Date:        10 October 1998
--| Project:     Janus/Warrior Combat Simulation for CAPS
--| Compiler:    ObjectAda for Windows Ver. 7.1.1 (Professional)
--| Description: This package describes basic functions and procedures
--|              involving event types in the Warrior Combat Simulation
--|              model.
------------------------------------------------------------------------
with ada.text_io;
PACKAGE BODY Event_Type_Pkg IS
```

```
-----------------------------------------------------------------
--| FUNCTION Get_Event_Time
--| Pre:     An unexecuted event exsist.
--| Post:    Start time for the event is returned.
-----------------------------------------------------------------
FUNCTION Get_Event_Time (Event: IN Event_Type'Class)
                                      RETURN Game_Time_Type IS
BEGIN -- Get_Event_Time
    RETURN Event.Time;
END Get_Event_Time;


-----------------------------------------------------------------
--| PROCEDURE Set_Event_Time
--| Pre:
--| Post:
-----------------------------------------------------------------
PROCEDURE Set_Event_Time (Event: IN OUT Event_Type'Class;
                                       Time: IN Game_Time_Type) IS
BEGIN -- Set_Event_Time
    Event.Time := Time;
END Set_Event_Time;


-----------------------------------------------------------------
--| FUNCTION Get_Object
--| Pre: An event exist.
--| Post: The object designated within the event is returned.
-----------------------------------------------------------------
FUNCTION Get_Object (Event: IN Event_Type'Class)
                                RETURN Simulation_Object_Ptr IS
BEGIN -- Get_Object
  RETURN Event.Object_Ptr;
END Get_Object;


-----------------------------------------------------------------
--| FUNCTION Get_Action
--| Pre: An event exist.
--| Post: The action on the object in the event is returned.
-----------------------------------------------------------------
FUNCTION Get_Action (Event: IN Event_Type'Class)
                                      RETURN Event_Action_Type IS
BEGIN -- Get_Action
  RETURN Event.Action;
END Get_Action;


-----------------------------------------------------------------
--| FUNCTION "<"
--| Pre: Two event types exist.
--| Post: The least valued event is returned.
-----------------------------------------------------------------
FUNCTION "<" (Left, Right: IN Event_Type_Ptr) RETURN Boolean IS
  Reply : Boolean;
BEGIN -- "<"
  IF Left.ALL.Time < Right.ALL.Time THEN
```

```
        Reply := True;
      ELSIF  Left.ALL.Time > Right.ALL.Time THEN
        Reply := False;
      ELSE
        Reply := ( Left.ALL.Action < Right.ALL.Action );
      END IF;
    RETURN Reply;
    END "<";

    -------------------------------------------------------------------
    --| FUNCTION Execute_Event
    --| Pre:     A move event has been extracted from the event queue
    --|          and needs to be executed.
    --| Post:    Move event is executed and time executed is returned.
    -------------------------------------------------------------------
    FUNCTION Execute_Event (Event: IN Event_Type)
                                      RETURN Game_Time_Type IS
    begin --
      ada.text_io.put_line("In the base execute event routine.");
      return 100;
    end execute_event;

    -------------------------------------------------------------------
    --| PROCEDURE Copy_Event
    --| Pre:     An move event exist.
    --| Post:    The move event is copied and a pointer to the copy is
    --|          returned.
    -------------------------------------------------------------------
    FUNCTION Copy_Event (Event: IN Event_Type) RETURN Event_Type_Ptr IS
    begin --
      ada.text_io.put_line("In the base copy routine");
      return null;
    end copy_event;

END Event_Type_Pkg;
```

34. EVENT_TYPE_PKG-END_SIM_PKG.ADS

```
-----------------------------------------------------------------------------
--| FileName:    Event_Type_Pkg.End_Sim_Pkg.ads
--| Author:      Julian Williams
--| Date:        10 October 1998
--| Project:     Janus/Warrior Combat Simulation for CAPS
--| Compiler:    ObjectAda for Windows Ver. 7.1.1 (Professional)
--| Description: This package describes basic functions and procedures
--|              involving event types in the Warrior Combat Simulation model.
-----------------------------------------------------------------------------
PACKAGE Event_Type_Pkg.End_Sim_Pkg IS

    TYPE End_Sim_Event_Type IS NEW Event_Type WITH PRIVATE;

    -------------------------------------------------------------------------
    --| FUNCTION Execute_Event
    --| Pre:     An end simulation event has been extracted from the event
    --|          queue and needs to be executed.
    --| Post:    End Simulation is executed and time executed is returned.
    -------------------------------------------------------------------------
    FUNCTION Execute_Event (Event: IN End_Sim_Event_Type) RETURN
                           Game_Time_Type;
```

```
    ---------------------------------------------------------------------------
    --| PROCEDURE Construct_Event
    --| Pre:      No event exist.
    --| Post:     A move event is constructed and the event is returned.
    ---------------------------------------------------------------------------
    FUNCTION Construct_Event (Object_Ptr: IN Simulation_Object_Ptr;
                              Time: IN Game_Time_Type)
                              RETURN Event_Type_Ptr;

    ---------------------------------------------------------------------------
    --| PROCEDURE Copy_Event
    --| Pre:      An event exist.
    --| Post:     The event is copied and the copy is returned.
    ---------------------------------------------------------------------------
    FUNCTION Copy_Event (Event: IN End_Sim_Event_Type) RETURN Event_Type_Ptr;

PRIVATE
    TYPE End_Sim_Event_Type IS NEW Event_Type WITH NULL RECORD;
END Event_Type_Pkg.End_Sim_Pkg;
```

35. EVENT_TYPE_PKG-END_SIM_PKG.ADB

```
-------------------------------------------------------------------------------
--| FileName:    Event_Type_Pkg.End_Sim_Pkg.adb
--| Author:      Julian Williams
--| Date:        10 October 1998
--| Project:     Janus/Warrior Combat Simulation for CAPS
--| Compiler:    ObjectAda for Windows Ver. 7.1.1 (Professional)
--| Description: This package describes basic functions and procedures
involving
--|              event types in the Warrior Combat Simulation model.
-------------------------------------------------------------------------------
WITH Warrior_1_Static_Schedulers;  USE Warrior_1_Static_Schedulers;
WITH Warrior_1_Dynamic_Schedulers; USE Warrior_1_Dynamic_Schedulers;
WITH Panel_Gui_3;
WITH warrior_event_monitor_task_pkg;

PACKAGE BODY Event_Type_Pkg.End_Sim_Pkg IS

    ---------------------------------------------------------------------------
    --| FUNCTION Execute_Event
    --| Pre:      An end simulation event has been extracted from the event
    --|           queue and needs to be executed.
    --| Post:     End simulation is executed and time executed is returned.
    ---------------------------------------------------------------------------
    FUNCTION Execute_Event (Event: IN End_Sim_Event_Type)
                                                  RETURN Game_Time_Type IS
        Time : Game_Time_Type := Event.Time;
    BEGIN -- Execute_Event
      Stop_Static_Schedule;
      Stop_Dynamic_Schedule;
      Panel_Gui_3.End_Simulation;
      warrior_event_monitor_task_pkg.warrior_event_monitor_task.end_task;
      RETURN Time;
    END Execute_Event;

    ---------------------------------------------------------------------------
    --| PROCEDURE Construct_Event
    --| Pre:      No event exist.
    --| Post:     A move event is constructed and the event is returned.
    ---------------------------------------------------------------------------
    FUNCTION Construct_Event (Object_Ptr: IN Simulation_Object_Ptr;
```

```
                                 Time: IN Game_Time_Type)
                                 RETURN Event_Type_Ptr IS
    Event: Event_Type_Ptr;

  BEGIN --
    Event := NEW End_Sim_Event_Type'(Action => EndSimulation,
                                Object_Ptr => Object_Ptr,
                                      Time => Time);

    RETURN Event;

  END Construct_Event;

  --------------------------------------------------------------------------
  --| PROCEDURE Copy_Event
  --| Pre:      An event exist.
  --| Post:     The event is copied and the copy is returned.
  --------------------------------------------------------------------------
  FUNCTION Copy_Event (Event: IN End_Sim_Event_Type) RETURN Event_Type_Ptr IS
      Copy: Event_Type_Ptr;
  BEGIN -- Copy_Event
      Copy := Construct_Event( Get_Object( Event ), Get_Event_Time( Event ));
    RETURN Copy;
  END Copy_Event;

END Event_Type_Pkg.End_Sim_Pkg;
```

36. EVENT_TYPE_PKG-MOVE_PKG.ADS

```
------------------------------------------------------------------------------
--| FileName:    Event_Type_Pkg.Move_Pkg.ads
--| Author:      Julian Williams
--| Date:        10 October 1998
--| Project:     Janus/Warrior Combat Simulation for CAPS
--| Compiler:    ObjectAda for Windows Ver. 7.1.1 (Professional)
--| Description: This package describes basic functions and procedures
--|              involving event types in the Warrior Combat Simulation model.
------------------------------------------------------------------------------
PACKAGE Event_Type_Pkg.Move_Pkg IS

    TYPE Move_Event_Type IS NEW Event_Type WITH PRIVATE;

    --------------------------------------------------------------------------
    --| FUNCTION Execute_Event
    --| Pre:     A move event has been extracted from the event queue and needs
    --|          to be executed.
    --| Post:    Move event is executed and time executed is returned.
    --------------------------------------------------------------------------
    FUNCTION Execute_Event (Event: IN Move_Event_Type) RETURN Game_Time_Type;

    --------------------------------------------------------------------------
    --| PROCEDURE Construct_Event
    --| Pre:     No event exist.
    --| Post:    A move event is constructed and the event is returned.
    --------------------------------------------------------------------------
    FUNCTION Construct_Event (Object_Ptr: IN Simulation_Object_Ptr;
                                Time: IN Game_Time_Type)
                                RETURN Event_Type_Ptr;
    --------------------------------------------------------------------------
    --| PROCEDURE Copy_Event
    --| Pre:     An move event exist.
    --| Post:    The move event is copied and a pointer to the copy is
    --|          returned.
```

```
    -------------------------------------------------------------------------
    FUNCTION Copy_Event (Event: IN Move_Event_Type) RETURN Event_Type_Ptr;


PRIVATE

    TYPE Move_Event_Type IS NEW Event_Type WITH NULL RECORD;

END Event_Type_Pkg.Move_Pkg;
```

37. EVENT_TYPE_PKG-MOVE_PKG.ADB

```
-----------------------------------------------------------------------------
--| FileName:    Event_Type_Pkg.Move_Pkg.adb
--| Author:      Julian Williams
--| Date:        10 October 1998
--| Project:     Janus/Warrior Combat Simulation for CAPS
--| Compiler:    ObjectAda for Windows Ver. 7.1.1 (Professional)
--| Description: This package describes basic functions and procedures
--|              involving event types in the Warrior Combat Simulation
--|              model.
-----------------------------------------------------------------------------
PACKAGE BODY Event_Type_Pkg.Move_Pkg IS

    -------------------------------------------------------------------------
    --| FUNCTION Execute_Event
    --| Pre:     An move event has been extracted from the event queue
    --|          and needs to be executed.
    --| Post:    Move event is executed and time executed is returned.
    -------------------------------------------------------------------------
    FUNCTION Execute_Event (Event: IN Move_Event_Type)
                                        RETURN Game_Time_Type IS
        Time: Game_Time_Type;
    BEGIN -- Execute_Event
        Time := Get_Event_Time(Event);
        Move_Update_Obj( Get_Object(Event).ALL, Time );
        RETURN Time;
    END Execute_Event;

    -------------------------------------------------------------------------
    --| PROCEDURE Construct_Event
    --| Pre:     No event exist.
    --| Post:    A move event is constructed and the event is returned.
    -------------------------------------------------------------------------
    FUNCTION Construct_Event (Object_Ptr: IN Simulation_Object_Ptr;
                              Time: IN Game_Time_Type)
                              RETURN Event_Type_Ptr IS
      Event: Event_Type_Ptr;

    BEGIN --
      Event := NEW Move_Event_Type'(Action => MoveUpdateObj,
                               Object_Ptr => Object_Ptr,
                                     Time => Time);

      RETURN Event;
    END Construct_Event;
```

```
---------------------------------------------------------------------
--| PROCEDURE Copy_Event
--| Pre:     An event exist.
--| Post:    The event is copied and the copy is returned.
---------------------------------------------------------------------
FUNCTION Copy_Event (Event: IN Move_Event_Type)
                                       RETURN Event_Type_Ptr IS
    Copy: Event_Type_Ptr;
BEGIN -- Copy_Event
  IF Get_Object( Event ) /= NULL THEN
    Copy := Construct_Event( Copy_Obj( Get_Object(Event).ALL ),
                                       Get_Event_Time( Event ) );
  ELSE
    Copy := Construct_Event( NULL, Get_Event_Time(Event) );
  END IF;
  RETURN Copy;
END Copy_Event;

END Event_Type_Pkg.Move_Pkg;
```

38. GAME_TIME_TYPE_PKG.ADS

```
package game_time_type_pkg is
  subtype game_time_type is integer range -1 .. integer'last;

  never: constant game_time_type := -1;

  function zero return game_time_type;

end game_time_type_pkg;
```

39. GAME_TIME_TYPE_PKG.ADB

```
package body game_time_type_pkg is

  function zero return game_time_type is
  begin
    return game_time_type(0);
  end zero;
end game_time_type_pkg;
```

40. GENERATED_TAE_EVENT_MONITOR_PKG.ADS

```
with Interfaces.C;
use Interfaces.C;

with linker_options_pragma_pkg;

package generated_tae_event_monitor_pkg is
      procedure generated_tae_event_monitor;
      pragma Import(C, generated_tae_event_monitor,
                           "generated_tae_event_monitor");

end generated_tae_event_monitor_pkg;
```

41. GET_RE_30_PKG.ADS

```
with replay_request_type_pkg; use replay_request_type_pkg;
package get_re_30_pkg is
  procedure get_re_30(replay_request : out replay_request_type);
  procedure record_input(replay_request : in replay_request_type);
```

```
  function has_new_input return boolean;
    -- True iff a user input has arrived
    -- since the last time this bubble was executed.
end get_re_30_pkg;
```

42. GET_RE_30_PKG.ADB

```
  with psdl_streams; use psdl_streams;
package body get_re_30_pkg is
  package replay_request_buffer is new
    sampled_buffer(replay_request_type);
  use replay_request_buffer;

  procedure get_re_30(replay_request : out replay_request_type) is
  begin
    -- Get the value from replay_request_buffer
    buffer.read(replay_request);
  end get_re_30;

  procedure record_input(replay_request : in replay_request_type) is
  begin
    -- Save the value in replay_request_buffer
    buffer.write(replay_request);
  end record_input;

  function has_new_input return boolean is
  begin
    -- Check status of replay_request_buffer
    return new_data;
  end has_new_input;
end get_re_30_pkg;
```

43. GET_ST_27_PKG.ADS

```
with statistics_request_type_pkg; use statistics_request_type_pkg;
package get_st_27_pkg is
  procedure get_st_27(statistics_request : out statistics_request_type);
  procedure record_input(statistics_request : in statistics_request_type);
  function has_new_input return boolean;
    -- True iff a user input has arrived
    -- since the last time this bubble was executed.
end get_st_27_pkg;
```

44. GET_ST_27_PKG.ADB

```
  with psdl_streams; use psdl_streams;
package body get_st_27_pkg is
  package statistics_request_buffer is new
    sampled_buffer(statistics_request_type);
  use statistics_request_buffer;

  procedure get_st_27(statistics_request : out statistics_request_type) is
  begin
    -- Get the value from statistics_request_buffer
    buffer.read(statistics_request);
  end get_st_27;

  procedure record_input(statistics_request : in statistics_request_type) is
  begin
    -- Save the value in statistics_request_buffer
    buffer.write(statistics_request);
  end record_input;
```

```
  function has_new_input return boolean is
  begin
    -- Check status of statistics_request_buffer
    return new_data;
  end has_new_input;
end get_st_27_pkg;
```

45. GET_USER_IN_21_PKG.ADS

```
with user_interaction_type_pkg; use user_interaction_type_pkg;
package get_user_in_21_pkg is
  procedure get_user_in_21(user_interaction : out user_interaction_type);
  procedure record_input(user_interaction : in user_interaction_type);
  function has_new_input return boolean;
    -- True iff a user input has arrived
    -- since the last time this bubble was executed.
end get_user_in_21_pkg;
```

46. GET_USER_IN_21_PKG.ADB

```
  with psdl_streams; use psdl_streams;
package body get_user_in_21_pkg is
  package user_interaction_buffer is new
    sampled_buffer(user_interaction_type);
  use user_interaction_buffer;

  procedure get_user_in_21(user_interaction : out user_interaction_type) is
  begin
    -- Get the value from user_interaction_buffer
    buffer.read(user_interaction);
  end get_user_in_21;

  procedure record_input(user_interaction : in user_interaction_type) is
  begin
    -- Save the value in user_interaction_buffer
    buffer.write(user_interaction);
  end record_input;

  function has_new_input return boolean is
  begin
    -- Check status of user_interaction_buffer
    return new_data;
  end has_new_input;
end get_user_in_21_pkg;
```

47. GET_X_65_PKG.ADS

```
package get_x_65_pkg is
  procedure get_x_65(new_x : out float);
  procedure record_input(new_x : in float);
  function has_new_input return boolean;
    -- True iff a user input has arrived
    -- since the last time this bubble was executed.
end get_x_65_pkg;
```

48. GET_X_65_PKG.ADB

```
  with psdl_streams; use psdl_streams;
package body get_x_65_pkg is
  package new_x_buffer is new
    sampled_buffer(float);
  use new_x_buffer;
```

```
  procedure get_x_65(new_x : out float) is
  begin
    -- Get the value from new_x_buffer
    buffer.read(new_x);
  end get_x_65;

  procedure record_input(new_x : in float) is
  begin
    -- Save the value in new_x_buffer
    buffer.write(new_x);
  end record_input;

  function has_new_input return boolean is
  begin
    -- Check status of new_x_buffer
    return new_data;
  end has_new_input;
end get_x_65_pkg;
```

49. GET_Y_68_PKG.ADS

```
package get_y_68_pkg is
  procedure get_y_68(new_y : out float);
  procedure record_input(new_y : in float);
  function has_new_input return boolean;
    -- True iff a user input has arrived
    -- since the last time this bubble was executed.
end get_y_68_pkg;
```

50. GET_Y_68_PKG.ADB

```
  with psdl_streams; use psdl_streams;
package body get_y_68_pkg is
  package new_y_buffer is new
    sampled_buffer(float);
  use new_y_buffer;

  procedure get_y_68(new_y : out float) is
  begin
    -- Get the value from new_y_buffer
    buffer.read(new_y);
  end get_y_68;

  procedure record_input(new_y : in float) is
  begin
    -- Save the value in new_y_buffer
    buffer.write(new_y);
  end record_input;

  function has_new_input return boolean is
  begin
    -- Check status of new_y_buffer
    return new_data;
  end has_new_input;
end get_y_68_pkg;
```

51. GUI_EVENT_MONITOR_18_PKG.ADS

```
package gui_event_monitor_18_pkg is
  procedure gui_event_monitor_18;
end gui_event_monitor_18_pkg;
```

52. GUI_EVENT_MONITOR_18_PKG.ADB

```
  with warrior_event_monitor_task_pkg;
  use warrior_event_monitor_task_pkg;
package body gui_event_monitor_18_pkg is
  procedure gui_event_monitor_18 is
  begin
    warrior_event_monitor_task.event_monitor_entry;
  end gui_event_monitor_18;
end gui_event_monitor_18_pkg;
```

53. INITIAL_SCENARIO_40_PKG.ADS

```
with scenario_type_pkg; use scenario_type_pkg;
package initial_scenario_40_pkg is
  procedure initial_scenario_40(scenario : out scenario_type;
                                first_time : in out boolean);
end initial_scenario_40_pkg;
```

54. INITIAL_SCENARIO_40_PKG.ADB

```
package body initial_scenario_40_pkg is
  procedure initial_scenario_40(scenario : out scenario_type;
                                first_time : in out boolean) is
  begin
    initialize_scenario(scenario);
    first_time := false;
  end initial_scenario_40;
end initial_scenario_40_pkg;
```

55. JAAWS_12_PKG.ADS

```
  with replay_request_type_pkg; use replay_request_type_pkg;
  with location_type_pkg; use location_type_pkg;
  with warrior_1_instantiations; use warrior_1_instantiations;
  with warrior_1_exceptions; use warrior_1_exceptions;

package jaaws_12_pkg is

  procedure jaaws_12(
    simulation_history: in event_type_sequence;
    replay_request: in out replay_request_type;
    replay_position: in out integer;
    replay: out location_type );
end jaaws_12_pkg;
```

56. JAAWS_12_PKG.ADB

```
  with simulation_object_pkg; use simulation_object_pkg;
  with event_type_pkg; use event_type_pkg;

package body jaaws_12_pkg is

  procedure jaaws_12(
    simulation_history: in event_type_sequence;
    replay_request: in out replay_request_type;
    replay_position: in out integer;
    replay: out location_type ) is
      -- Precondition: not is_empty(simulation_history)
      -- Precondition: 1 <= replay_position <= length(simulation_history)
      i: integer;
      e: event_type_ptr;
      o: simulation_object_ptr;
  begin
    -- replay_position = previous snapshot location or 1
```

```
    -- Set replay to the previous snapshot.
    e := fetch(simulation_history, replay_position);
    if get_action(e.all) = MoveUpdateObj then
       o := get_object(e.all);
       replay := get_location(o.all);
    else -- the previous position is not at a move event
         replay := origin;
    end if;

    -- Set i to the tentative new replay position
    if replay_request = on then -- reset to the beginning
       replay_request := off;
       i := 1;
       -- e := fetch(simulation_history, i);
       -- o := get_object(e.all);
       -- replay := get_location(o.all);
       -- replay_position := i;
    elsif replay_position < length(simulation_history) then
          i := replay_position + 1;
    else i := replay_position; -- Already at the end, stay there.
    end if;

    -- Advance i to the location of the next move event if there is one.
    -- Invariant: 1 <= i <= length(simulation_history)

    e := fetch(simulation_history, i);
    while get_action(e.all) /= MoveUpdateObj loop
       if i < length(simulation_history) then
          i := i + 1;
          e := fetch(simulation_history, i);
       else -- There is no next move event, stay at the previous position.
            -- i is at the last simulation history event and it is not
            -- a MoveUpdateObj event, so do nothing
            -- replay_position maintains the old value
            -- replay maintains either old value or origin
            return;
       end if;
    end loop;
    -- i is at a new MoveUpdateObj event position
    o := get_object(e.all);
    replay := get_location(o.all);
    replay_position := i;
  end jaaws_12;
end jaaws_12_pkg;
```

57. LINKER_OPTIONS_PRAGMA_PKG.ADS

```
package linker_options_pragma_pkg is
   pragma Linker_Options("warrior_tae.c");
   pragma Linker_Options("warrior_pan_gui_3.c");
   pragma Linker_Options("warrior_creat_init.c");
   pragma Linker_Options("warrior_init_pan.c");
   pragma Linker_Options("-I/local/tae/include");
   pragma Linker_Options("/local/tae/lib/sun4/libwpt.a");
   pragma Linker_Options("/local/tae/Xtae/lib/sun4/libXtae.a");
   pragma Linker_Options("/local/tae/Xtae/lib/sun4/libddo.a");
   pragma Linker_Options("/local/tae/lib/sun4/libwmw.a");
   pragma Linker_Options("/local/tae/Xtae/lib/sun4/libIV.a");
   pragma Linker_Options("/local/tae/Xtae/lib/sun4/libxterm.a");
   pragma Linker_Options("/usr/lib/libXm.a");
   pragma Linker_Options("/usr/lib/libXt.a");
   pragma Linker_Options("/usr/lib/libXmu.a");
   pragma Linker_Options("/usr/lib/libXext.a");
```

```
   pragma Linker_Options("/usr/lib/libX11.a");
   pragma Linker_Options("/local/tae/lib/sun4/libtaec.a");
   pragma Linker_Options("/local/tae/lib/sun4/libtae.a");
   pragma Linker_Options("/usr/lib/libtermlib.a");
   pragma Linker_Options("/usr/lib/libm.a");
   pragma Linker_Options("/usr/local/lib/libcxx.a");
end linker_options_pragma_pkg;
```

58. LOCATION_TYPE_PKG.ADS

```
package location_type_pkg is

   type Location_Type is record
         X : Float := 0.0;
         Y : Float := 0.0;
         Z : Float := 0.0;
      end record;

   FUNCTION "+" (L1,L2 : Location_Type) RETURN Location_Type;

   FUNCTION "-" (L1,L2 : Location_Type) RETURN Location_Type;

   FUNCTION "*" (C : Float; L : Location_Type) RETURN Location_Type;

   FUNCTION Length (L : Location_Type) RETURN Float;

   FUNCTION "=" (L1,L2 : Location_Type) RETURN Boolean;

   FUNCTION Get_X (L : Location_Type) RETURN Float;

   FUNCTION Get_Y (L : Location_Type) RETURN Float;

   FUNCTION Origin RETURN Location_Type;

   FUNCTION To_Location(X, Y: Float) RETURN Location_Type;
end location_type_pkg;
```

59. LOCATION_TYPE_PKG.ADB

```
-- -------------------------------------------------------------------------
-- File Name:    Location_Type_Pkg.Adb
-- -------------------------------------------------------------------------
WITH Ada.Numerics.Elementary_Functions;  --Used for Square Root
USE  Ada.Numerics.Elementary_Functions;

PACKAGE BODY Location_Type_Pkg IS

   FUNCTION "+" (L1,L2 : Location_Type) RETURN Location_Type IS
      BEGIN
         RETURN (X=> L1.X + L2.X, Y=> L1.Y + L2.Y, Z=> L1.Z + L2.Z);
      END;

   FUNCTION "-" (L1,L2 : Location_Type) RETURN Location_Type IS
      BEGIN
         RETURN (X=> L1.X - L2.X, Y=> L1.Y - L2.Y, Z=> L1.Z - L2.Z);
      END;

   FUNCTION "*" (C : Float; L : Location_Type) RETURN Location_Type IS
      BEGIN
         RETURN (X=> C * L.X, Y=> C * L.Y, Z=> C * L.Z);
      END;

   FUNCTION Length (L : Location_Type) RETURN Float IS
```

```
      BEGIN
         RETURN Sqrt((L.X * L.X) + (L.Y * L.Y) + (L.Z * L.Z));
      END;

   FUNCTION "=" (L1,L2 : Location_Type) RETURN Boolean IS
      BEGIN
         RETURN (L1.X=L2.X AND L1.Y=L2.Y AND L1.Z=L2.Z);
      END;

   FUNCTION Get_X (L : Location_Type) RETURN Float IS
      BEGIN
         RETURN L.X;
      END;

   FUNCTION Get_Y (L : Location_Type) RETURN Float IS
      BEGIN
         RETURN L.Y;
      END;

   FUNCTION Origin RETURN Location_Type IS
         L : Location_Type:=(X=>0.0, Y=>0.0, Z=>0.0);
      BEGIN
         RETURN L;
      END;

   FUNCTION To_Location(X, Y: Float) RETURN Location_Type IS
         L : Location_Type:=(X=>X, Y=>Y, Z=>0.0);
      BEGIN
         RETURN L;
      END;

END Location_Type_Pkg;
```

60. LOOKAHEAD_STREAM_PKG.ADS

```
with io_exceptions;
  with delimiter_pkg; use delimiter_pkg;
package lookahead_stream_pkg is
  function token return character;
    -- Returns the next non-blank character without removing it.
    -- Raises constraint_error if no more tokens in the buffer.

  procedure skip_char; -- removes the current character.

  end_error: exception renames io_exceptions.end_error;
    -- Attempt to read past end of file.
end lookahead_stream_pkg;
```

61. LOOKAHEAD_STREAM_PKG.ADB

```
  with text_io; use text_io;
package body lookahead_stream_pkg is
  blank: constant delimiter_array := initialize_delimiter_array;
  buffer: character;
  empty: boolean := true;
  -- (~empty => buffer is the next character in the stream).

  function peek return character is
  begin
    if empty then get(buffer); empty := false; end if;
    return buffer;
  end peek;
```

```
  function token return character is
   -- Blank is a constant array, see top of package body.
  begin
     -- Advance the lookahead stream to a non-blank character.
   while blank(peek) loop skip_char; end loop;
     -- Return the character without removing it from the stream.
   return peek;
  end token;

  procedure skip_char is
  begin
    if empty then get(buffer);  -- Read and discard next character.
    else empty := true; end if;  -- Discard character in the buffer.
  end skip_char;
end lookahead_stream_pkg;
```

62. NATURAL_SET_IO_PKG.ADS

```
with natural_set_pkg;
with text_io;
with integer_io;

package natural_set_io_pkg is

  procedure put(ns: in natural_set_pkg.set);

end natural_set_io_pkg;
```

63. NATURAL_SET_IO_PKG.ADB

```
package body natural_set_io_pkg is

  package natural_io is new text_io.integer_io(NATURAL);

  procedure put_n(i: in natural) is
  begin
    natural_io.put(i);
  end put_n;

  procedure mput is new natural_set_pkg.generic_put(put_n);

  procedure put(ns: in natural_set_pkg.set) is
  begin
    mput(ns);
  end put;

end natural_set_io_pkg;
```

64. NATURAL_SET_PKG.ADS

```
with set_pkg;

package natural_set_pkg is NEW set_pkg(NATURAL,"=");
```

65. PANEL_GUI_3.ADS

```
with Interfaces.C;
use Interfaces.C;

with statistics_type_pkg;
use statistics_type_pkg;
with location_type_pkg;
use location_type_pkg;
```

```
package panel_gui_3 is

	-- procedures for calling c routines to display info to GUI

	procedure display_st_31(statistics: in statistics_type);

	procedure display_re_37(replay: in location_type);


	-- procedures to be called by the c routines to handle push button events

	procedure set_user_interaction;
	pragma Export(C, set_user_interaction, "set_user_interaction");

	procedure set_statistics_request;
	pragma Export(C, set_statistics_request, "set_statistics_request");

	procedure set_replay_request;
	pragma Export(C, set_replay_request, "set_replay_request");

	procedure set_new_plan;
	pragma Export(C, set_new_plan, "set_new_plan");

	procedure end_simulation;
	pragma Import(C, end_simulation, "end_simulation");

	procedure set_x(x : in double);
	pragma Export(C, set_x, "set_x");

	procedure set_y(y : in double);
	pragma Export(C, set_y, "set_y");

	procedure initialize_gui;
	pragma Import(C, initialize_gui, "initialize_gui");

end panel_gui_3;
```

66. PANEL_GUI_3.ADB

```
with Interfaces.C;
use Interfaces.C;

with statistics_type_pkg;
use statistics_type_pkg;
with location_type_pkg;
use location_type_pkg;
with replay_request_type_pkg;
use replay_request_type_pkg;
with statistics_request_type_pkg;
use statistics_request_type_pkg;
with user_interaction_type_pkg;
use user_interaction_type_pkg;

with get_user_in_21_pkg;
with get_st_27_pkg;
with get_re_30_pkg;
with get_x_65_pkg;
with get_y_68_pkg;
with enter_new_plan_75_pkg;

with text_io;
with ada.float_text_io;
use ada.float_text_io;
```

```
package body panel_gui_3 is

        procedure display_fuel_consumption(c: in double);
        pragma Import(C, display_fuel_consumption, "display_fuel_consumption");

        procedure display_xloc(x: in double);
        pragma Import(C, display_xloc, "display_xloc");

        procedure display_yloc(y: in double);
        pragma Import(C, display_yloc, "display_yloc");

        procedure display_mover(x, y: in double);
        pragma Import(C, display_mover, "display_mover");

        procedure display_st_31(statistics: statistics_type) is
                d : double := double(statistics_type_pkg.convert(statistics));
        begin

                display_fuel_consumption(d);
        end display_st_31;

        procedure display_re_37(replay: location_type) is
                x, y : double;
        begin
                -- need code to extract x, y from location type;
                -- set x, y to dummy value 5.0, -5.0 for the time being
                x := double(location_type_pkg.get_x(replay));
                y := double(location_type_pkg.get_y(replay));
                display_xloc(x);
                display_yloc(y);
                display_mover(x, y);
        end display_re_37;

        procedure set_user_interaction is
                v : user_interaction_type
                        := user_interaction_type_pkg.stop_simulation;
        begin
                get_user_in_21_pkg.record_input(v);
        end set_user_interaction ;

        procedure set_statistics_request is
                v : statistics_request_type := statistics_request_type_pkg.on;
        begin
                get_st_27_pkg.record_input(v);
        end set_statistics_request ;

        procedure set_replay_request is
                v : replay_request_type := replay_request_type_pkg.on;
        begin
                get_re_30_pkg.record_input(v);
        end set_replay_request;

        procedure set_new_plan is
        begin
                enter_new_plan_75_pkg.record_input(true);
        end set_new_plan;

         procedure set_x(x : in double) is
         begin
                get_x_65_pkg.record_input(float(x));
         end set_x;
```

```
        procedure set_y(y : in double) is
        begin
             get_y_68_pkg.record_input(float(y));
        end set_y;

end panel_gui_3;
```

67. POST_PROCESSOR_6_PKG.ADS

```
with statistics_request_type_pkg; use statistics_request_type_pkg;
  with statistics_type_pkg; use statistics_type_pkg;
  with warrior_1_instantiations; use warrior_1_instantiations;
  with warrior_1_exceptions; use warrior_1_exceptions;

package post_processor_6_pkg is

  procedure post_processor_6(
    statistics_request: in statistics_request_type;
    simulation_history: in event_type_sequence;
    statistics: out statistics_type );
end post_processor_6_pkg;
```

68. POST_PROCESSOR_6_PKG.ADB

```
  with simulation_object_pkg; use simulation_object_pkg;
  with event_type_pkg; use event_type_pkg;
package body post_processor_6_pkg is

  procedure post_processor_6(
    statistics_request: in statistics_request_type;
    simulation_history: in event_type_sequence;
    statistics: out statistics_type ) is
      o: simulation_object_ptr;
      e: event_type_ptr;
      fuel_used: float := 0.0;
  begin
    -- This version assumes a vehicle never refuels
    for i IN 1 .. length(simulation_history) loop
        e := fetch(simulation_history, i);
        if get_action(e.all) = MoveUpdateObj then
           o := get_object(e.all);
           fuel_used := get_fuel_used(o.all);
        end if;
    end loop;
    statistics := convert(fuel_used);
  end post_processor_6;
end post_processor_6_pkg;
```

69. REPLAY_REQUEST_TYPE_PKG.ADS

```
package replay_request_type_pkg is
  type replay_request_type is private;

  function on return replay_request_type;

  function off return replay_request_type;

private
  type replay_request_type is new boolean;
end replay_request_type_pkg;
```

70. REPLAY_REQUEST_TYPE_PKG.ADB

```
package body replay_request_type_pkg is

  function on return replay_request_type is
  begin
    return true;
  end on;

  function off return replay_request_type is
  begin
    return false;
  end off;
end replay_request_type_pkg;
```

71. SCENARIO_TYPE_PKG.ADS

```
with Simulation_Object_Pkg; use Simulation_Object_Pkg;
package scenario_type_pkg is
  type scenario_type is private;

  PROCEDURE Initialize_Scenario(Sc1 : OUT Scenario_Type);

  function empty_scenario return scenario_type;

  function is_empty(SC1 : Scenario_Type) return boolean;

  function get_unit(SC1 : Scenario_Type) RETURN Simulation_Object_Ptr;

private
   type scenario_type is record
         Scenario_Name : String(1..20) := "empty scenario      ";
         Unit          : Simulation_Object_Ptr := NULL;  --Now only 1 obj,
could be List
         --Terrain     : Terrain_Type;
         --Weather     : Weather_Type;
      end record;

end scenario_type_pkg;
```

72. SCENARIO_TYPE_PKG.ADB

```
WITH Simulation_Object_Pkg.Ground_Object_Pkg.Tank_Pkg;
  USE  Simulation_Object_Pkg.Ground_Object_Pkg.Tank_Pkg;

PACKAGE BODY Scenario_Type_Pkg IS
  function get_unit(SC1 : Scenario_Type) RETURN Simulation_Object_Ptr IS
     BEGIN
        RETURN SC1.Unit;
     END;

  function empty_scenario return scenario_type is
       dummy : scenario_type;
  begin
    return dummy;
  end empty_scenario;

  function is_empty(SC1 : Scenario_Type) return boolean is
  begin
    return SC1.Unit = null;
  end is_empty;

   PROCEDURE Initialize_Scenario(Sc1 : OUT Scenario_Type) IS
```

```
      BEGIN
         Sc1.Scenario_Name:="Scenario One          ";
         Sc1.Unit := Construct_Obj(Scheduled => False,
                                   Name      => "M1A1                    ",
                                   Symbol    => 1,
                                   Force     => 1,
                                   Move_Period => 10,
                                   Active    => True,
                                   Location_x => -100.0,
                                   Location_y => -100.0,
                                   Destination_x => 3000.0,
                                   Destination_y => 3000.0,
                                   Speed =>  10.0,
                                   Max_Speed =>  25.0,
                                   Fuel => 500.0,
                                   Consumption =>   0.36);
      END;

END Scenario_Type_Pkg;
```

73. SEQUENCE_PKG.ADS

```
with natural_set_pkg;
with text_io;
use  text_io;

generic
  type t is private;

package sequence_pkg is
  type    sequence     is private;
  subtype natural_set is natural_set_pkg.set;
  function empty return sequence;
  procedure empty(ss : out sequence);
  function add(x : t; s : sequence) return sequence;
  procedure add(x : in t; s : in sequence; ss : out sequence);
  generic
    with function equal(x, y : t) return boolean is <>;
  function remove(x : t; s : sequence) return sequence;
  function append(s1, s2 : sequence) return sequence;
  procedure append(s1, s2 : in sequence; ss : out sequence);
  function fetch(s : sequence; n : natural) return t;
  procedure fetch(s : in sequence; n : in natural; tt : out t);
  function fetch(s1 : sequence; low, high : natural) return sequence;
  procedure fetch(s1 : in sequence; low, high : in natural; ss : out sequence);
  function length(s : sequence) return natural;
  procedure length(s : in sequence; nn : out natural);
  function domain(s : sequence) return natural_set;
  procedure domain(s : in sequence; ns : out natural_set);
  generic
    with function equal(x, y : t) return boolean is <>;
  function is_in(x : t; s : sequence) return boolean;
  generic
    with function equal(x, y : t) return boolean is <>;
  function part_of(s1, s2 : sequence) return boolean;
  generic
    with function equal(x, y : t) return boolean is <>;
  function generic_equal(s1, s2 : sequence) return boolean;
  generic
    with function "<" (x, y : t) return boolean is <>;
  function less_than(s1, s2 : sequence) return boolean;
  generic
```

```
    with function "<" (x, y : t) return boolean is <>;
    with function equal(x, y : t) return boolean is <>;
  function less_than_or_equal(s1, s2 : sequence) return boolean;
  generic
    with function "<" (x, y : t) return boolean is <>;
  function greater_than(s1, s2 : sequence) return boolean;
  generic
    with function "<" (x, y : t) return boolean is <>;
    with function equal(x, y : t) return boolean is <>;
  function greater_or_equal(s1, s2 : sequence) return boolean;
  generic
    with function equal(x, y : t) return boolean is <>;
  function subsequence(s1, s2 : sequence) return boolean;
  generic
    with function "<" (x, y : t) return boolean is <>;
    with function successor(x : t) return t;
  function interval(x1, x2 : t) return sequence;
  generic
    type et is private;
    type st is private;
    with function f(x : et) return t;
    with function length(s : st) return natural is <>;
    with function fetch(s : st; n : natural) return et is <>;
  function apply(s1 : st) return sequence;
  generic
    with function f(x, y : t) return t;
    identity : t;
  function reduce(s : sequence) return t;
  generic
    with function f(x, y : t) return t;
  function reduce1(s : sequence) return t;
  generic
    with procedure generate(x1 : in t);
  procedure scan(s : sequence);
  generic
    with function input return t is <>;
  function generic_input return sequence;
  generic
    with function input return t is <>;
  function generic_file_input(file : file_type) return sequence;
  generic
    with procedure put(item : in t) is <>;
  procedure generic_put(item : in sequence);
  generic
    with procedure put(item : in t) is <>;
  procedure generic_file_put(file : in file_type; item : in sequence);
  bounds_error              : exception;
  empty_reduction_undefined : exception;
private
  type sequence_record;
  type sequence_ptr   is access sequence_record;
  type sequence is record
    p : sequence_ptr := null;
  end record;
end sequence_pkg;
```

74. SEQUENCE_PKG.ADB

```
with lookahead_stream_pkg;

use  lookahead_stream_pkg;
```

```
package body sequence_pkg is
  use  natural_set_pkg;

  type sequence_record is record
    value : t;
    rest  : sequence;
  end record;

  function empty return sequence is
    s : sequence;

  begin
    return s;
  end empty;

  procedure empty(ss : out sequence) is
  begin
    ss := empty;
  end empty;

  function add(x : t; s : sequence) return sequence is
    s1 : sequence;

  begin
    if s = empty then
      s1.p := new sequence_record'(value => x, rest => s);

    else
      s1.p := new sequence_record'(value => s.p.value,
                                   rest => add(x, s.p.rest));
    end if;
    return s1;
  end add;

  procedure add(x : in t; s : in sequence; ss : out sequence) is
  begin
    ss := add(x, s);
  end add;

  function remove(x : t; s : sequence) return sequence is
    ss      : sequence;
    local_x : t := x;

  begin    -- begin generator loop
    declare
      exit_from_generator_loop : exception;
      procedure generator_loop_body(y : t) is
      begin

      if not equal(local_x, y) then
        ss := add(y, ss);
      end if;
      end generator_loop_body;
      procedure execute_generator_loop is new
                              scan(generator_loop_body);
    begin
```

```
    execute_generator_loop(s);
  exception
    when exit_from_generator_loop =>
    null;
  end;          -- of generator loop
  return ss;
end remove;

function append(s1, s2 : sequence) return sequence is
  ss : sequence;

begin    -- begin generator loop
  declare
    exit_from_generator_loop : exception;
    procedure generator_loop_body(x : t) is
    begin
    ss := add(x, ss);
    end generator_loop_body;
    procedure execute_generator_loop is new
                            scan(generator_loop_body);
  begin
    execute_generator_loop(s1);
  exception
    when exit_from_generator_loop =>
    null;
  end;                                  -- of generator loop
  declare                               -- begin generator loop
    exit_from_generator_loop : exception;
    procedure generator_loop_body(x : t) is
    begin
    ss := add(x, ss);
    end generator_loop_body;
    procedure execute_generator_loop is new
                                  scan(generator_loop_body);
  begin
    execute_generator_loop(s2);
  exception
    when exit_from_generator_loop =>
    null;
  end;    -- of generator loop
  return ss;
end append;

procedure append(s1, s2 : in sequence; ss : out sequence) is
begin
  ss := append(s1, s2);
end append;

function fetch(s : sequence; n : natural) return t is
  index : natural := 1;

begin    -- begin generator loop
  declare
    generator_loop_return_value : t;
    return_from_generator_loop  : exception;
    exit_from_generator_loop    : exception;
    procedure generator_loop_body(y : t) is
```

```
    begin

    if index = n then
      generator_loop_return_value := y;
      raise return_from_generator_loop;
    end if;
    index := index + 1;
    end generator_loop_body;
    procedure execute_generator_loop is new
                                scan(generator_loop_body);
  begin
    execute_generator_loop(s);
  exception
    when exit_from_generator_loop =>
    null;
    when return_from_generator_loop =>
    return generator_loop_return_value;
  end;                                -- of generator loop
  raise bounds_error;
end fetch;

procedure fetch(s : in sequence; n : in natural; tt : out t) is
begin
  tt := fetch(s, n);
end fetch;

function fetch(s1 : sequence; low, high : natural) return sequence is
  ss : sequence;

begin
  for i in low .. high loop
    ss := add(fetch(s1, i), ss);
  end loop;
  return ss;
end fetch;

procedure fetch(s1 : in sequence; low, high : in natural;
                                            ss : out sequence) is
begin
  ss := fetch(s1, low, high);
end fetch;

function length(s : sequence) return natural is
  index : natural := 0;

begin    -- begin generator loop
  declare
    exit_from_generator_loop : exception;
    procedure generator_loop_body(y : t) is
    begin
    index := index + 1;
    end generator_loop_body;
    procedure execute_generator_loop is new
                                scan(generator_loop_body);
  begin
    execute_generator_loop(s);
  exception
```

```
    when exit_from_generator_loop =>
    null;
  end;                              -- of generator loop
  return index;
end length;

procedure length(s : in sequence; nn : out natural) is
begin
  nn := length(s);
end length;

function domain(s : sequence) return natural_set is
  ns : natural_set := empty;

begin
  for i in 1 .. length(s) loop
    ns := add(i, ns);
  end loop;
  return ns;
end domain;

procedure domain(s : in sequence; ns : out natural_set) is
begin
  ns := domain(s);
end domain;

function is_in(x : t; s : sequence) return boolean is
  local_x : t := x;

begin    -- begin generator loop
  declare
    generator_loop_return_value : boolean;
    return_from_generator_loop  : exception;
    exit_from_generator_loop    : exception;
    procedure generator_loop_body(y : t) is
    begin

    if equal(local_x, y) then
      generator_loop_return_value := true;
      raise return_from_generator_loop;
    end if;
    end generator_loop_body;
    procedure execute_generator_loop is new
                            scan(generator_loop_body);
  begin
    execute_generator_loop(s);
  exception
    when exit_from_generator_loop =>
    null;
    when return_from_generator_loop =>
    return generator_loop_return_value;
  end;                              -- of generator loop
  return false;
end is_in;

function part_of(s1, s2 : sequence) return boolean is
  n : natural := 0;
```

```
  function matches_at(s1, s2 : sequence; n : natural)
                                              return boolean is
    i : natural := 0;

  begin
    while i < length(s1) loop
    if equal(fetch(s1, i + 1), fetch(s2, n + i)) then
      i := i + 1;

    else
      return false;
    end if;
    end loop;
    return true;
  end matches_at;

begin
  while n + length(s1) <= length(s2) loop
    if matches_at(s1, s2, n + 1) then
    return true;

    else
    n := n + 1;
    end if;
  end loop;
  return false;
end part_of;

function generic_equal(s1, s2 : sequence) return boolean is
  i        : natural  := 1;
  local_s2 : sequence := s2;

begin
  if length(s1) /= length(s2) then
    return false;
  end if;
  declare     -- begin generator loop
    generator_loop_return_value : boolean;
    return_from_generator_loop  : exception;
    exit_from_generator_loop    : exception;
    procedure generator_loop_body(x : t) is
    begin

    if not equal(x, fetch(local_s2, i)) then
      generator_loop_return_value := false;
      raise return_from_generator_loop;
    end if;
    i := i + 1;
    end generator_loop_body;
    procedure execute_generator_loop is new
                            scan(generator_loop_body);
  begin
    execute_generator_loop(s1);
  exception
    when exit_from_generator_loop =>
    null;
```

```
    when return_from_generator_loop =>
    return generator_loop_return_value;
  end;    -- of generator loop
  return true;
end generic_equal;

function less_than(s1, s2 : sequence) return boolean is
  i        : natural  := 1;
  y        : t;
  local_s2 : sequence := s2;

begin    -- begin generator loop
  declare
    generator_loop_return_value : boolean;
    return_from_generator_loop  : exception;
    exit_from_generator_loop    : exception;
    procedure generator_loop_body(x : t) is
    begin
    y := fetch(local_s2, i);

    if x < y then
      generator_loop_return_value := true;
      raise return_from_generator_loop;

    elsif y < x then
      generator_loop_return_value := false;
      raise return_from_generator_loop;
    end if;
    i := i + 1;
    end generator_loop_body;
    procedure execute_generator_loop is new
                               scan(generator_loop_body);
  begin
    execute_generator_loop(s1);
  exception
    when exit_from_generator_loop =>
    null;
    when return_from_generator_loop =>
    return generator_loop_return_value;
  end;                                        -- of generator loop
  return (length(s1) < length(s2));
end less_than;

function less_than_or_equal(s1, s2 : sequence) return boolean is
  function lt is new less_than;
  function seq_equal is new generic_equal(equal);

begin
  return lt(s1, s2) or else seq_equal(s1, s2);
end less_than_or_equal;

function greater_than(s1, s2 : sequence) return boolean is
  function lt is new less_than;

begin
  return lt(s2, s1);
end greater_than;
```

```
function greater_or_equal(s1, s2 : sequence) return boolean is
  function lt is new less_than;
  function seq_equal is new generic_equal(equal);

begin
  return lt(s2, s1) or else seq_equal(s1, s2);
end greater_or_equal;

function subsequence(s1, s2 : sequence) return boolean is
  i        : natural  := 0;
  local_s1 : sequence := s1;

begin
  if s1 = empty then
    return (true);
  end if;
  declare     -- begin generator loop
    generator_loop_return_value : boolean;
    return_from_generator_loop  : exception;
    exit_from_generator_loop    : exception;
    procedure generator_loop_body(x : t) is
    begin

    if equal(x, fetch(local_s1, i + 1)) then
      i := i + 1;

      if i = length(local_s1) then
        generator_loop_return_value := (true);
        raise return_from_generator_loop;
      end if;
    end if;
    end generator_loop_body;
    procedure execute_generator_loop is new
                            scan(generator_loop_body);
  begin
    execute_generator_loop(s2);
  exception
    when exit_from_generator_loop =>
    null;
    when return_from_generator_loop =>
    return generator_loop_return_value;
  end;    -- of generator loop
  return false;
end subsequence;

function interval(x1, x2 : t) return sequence is
  ss : sequence;
  y  : t := x1;

begin
  while (y < x2) loop
    ss := add(y, ss);
    y := successor(y);
  end loop;

  if y = x2 then
    ss := add(y, ss);
```

```
  end if;
  return ss;
end interval;

function apply(s1 : st) return sequence is
  ss : sequence;

begin
  for i in 1 .. length(s1) loop
    ss := add(f(fetch(s1, i)), ss);
  end loop;
  return ss;
end apply;

function reduce(s : sequence) return t is
  x : t := identity;

begin    -- begin generator loop
  declare
    exit_from_generator_loop : exception;
    procedure generator_loop_body(y : t) is
    begin
    x := f(y, x);
    end generator_loop_body;
    procedure execute_generator_loop is new
                            scan(generator_loop_body);
  begin
    execute_generator_loop(s);
  exception
    when exit_from_generator_loop =>
    null;
  end;                         -- of generator loop
  return x;
end reduce;

function reduce1(s : sequence) return t is
  x : t;
  i : natural := 1;

begin
  if s = empty then
    raise empty_reduction_undefined;
  end if;
  x := fetch(s, 1);
  declare    -- begin generator loop
    exit_from_generator_loop : exception;
    procedure generator_loop_body(y : t) is
    begin

    if i > 1 then
      x := f(y, x);
    end if;
    i := i + 1;
    end generator_loop_body;
    procedure execute_generator_loop is new
                            scan(generator_loop_body);
  begin
```

```
    execute_generator_loop(s);
  exception
    when exit_from_generator_loop =>
    null;
  end;    -- of generator loop
  return x;
end reduce1;

procedure scan(s : sequence) is
  ss : sequence := s;

begin
  while ss.p /= null loop
    generate(ss.p.value);
    ss := ss.p.rest;
  end loop;
end scan;

function generic_input return sequence is
  x  : t;
  ss : sequence;

begin
  if token /= ascii.l_bracket then
    raise data_error;
  end if;
  skip_char;
  while token /= ascii.r_bracket loop
    x := input;
    ss := add(x, ss);

    if token = ',' then
    skip_char;

    elsif token /= ascii.r_bracket then
    raise data_error;
    end if;
  end loop;
  skip_char;
  return ss;
exception
  when others =>
    raise data_error;
end generic_input;

function generic_file_input(file : file_type) return sequence is
  function get_sequence is new generic_input;
  s : sequence;

begin
  set_input(file);
  s := get_sequence;
  set_input(standard_input);
  return s;
end generic_file_input;

procedure generic_put(item : in sequence) is
```

```
  begin
    put(ascii.l_bracket);

    if length(item) >= 1 then
      put(fetch(item, 1));
    end if;
    for i in 2 .. length(item) loop
      put(", ");
      put(fetch(item, i));
    end loop;
    put(ascii.r_bracket);
  end generic_put;

  procedure generic_file_put(file : in file_type;
                                         item : in sequence) is
    procedure put_sequence is new generic_put;

  begin
    set_output(file);
    put_sequence(item);
    set_output(standard_output);
  end generic_file_put;
end sequence_pkg;
```

75. SET_PKG.ADS

```
with text_io;
use  text_io;

generic
  type t is private;
  with function t_equal(x, y : t) return boolean is "=";

package set_pkg is
  type set is private;
  function empty return set;
  procedure empty(ss : out set);
  function add(x : t; s : set) return set;
  procedure add(x : in t; s : in set; ss : out set);
  function remove(x : t; s : set) return set;
  procedure remove(x : in t; s : in set; ss : out set);
  function is_in(x : t; s : set) return boolean;
  procedure is_in(x : in t; s : in set; bb : out boolean);
  function union(s1, s2 : set) return set;
  procedure union(s1, s2 : in set; ss : out set);
  function difference(s1, s2 : set) return set;
  procedure difference(s1, s2 : in set; ss : out set);
  function intersection(s1, s2 : set) return set;
  procedure intersection(s1, s2 : in set; ss : out set);
  function choose(s : set) return t;
  procedure choose(s : in set; tt : out t);
  function size(s : set) return natural;
  procedure size(s : in set; nn : out natural);
  function equal(s1, s2 : set) return boolean;
  procedure equal(s1, s2 : in set; bb : out boolean);
  function subset(s1, s2 : set) return boolean;
  procedure subset(s1, s2 : in set; bb : out boolean);
  generic
    with function "<" (x, y : t) return boolean is <>;
    with function successor(x : t) return t;
```

```
  function interval(x1, x2 : in t) return set;
  generic
    type et is private;
    type st is private;
    with function f(x : t) return et is <>;
    with function empty return st is <>;
    with function add(x : et; s : st) return st is <>;
  function apply(s : set) return st;
  generic
    with function f(x, y : t) return t;
    identity : t;
  function reduce(s : set) return t;
  generic
    with function f(x, y : t) return t;
  function reduce1(s : set) return t;
  generic
    with procedure generate(x1 : in t);
  procedure scan(s : set);
  empty_set                  : exception;
  empty_reduction_undefined : exception;
  generic
    with function input return t is <>;
  function generic_input return set;
  generic
    with function input return t is <>;
  function generic_file_input(file : in file_type) return set;
  generic
    with procedure put(item : in t) is <>;
  procedure generic_put(item : in set);
  generic
    with procedure put(item : in t) is <>;
  procedure generic_file_put(file : in file_type; item : in set);
private
  type set_record;
  type set_ptr   is access set_record;
  type set is record
    p : set_ptr := null;
  end record;
end set_pkg;
```

76. SET_PKG.ADB.

```
with lookahead_stream_pkg;

use  lookahead_stream_pkg;


package body set_pkg is
  type set_record is record
    value : t;
    rest  : set;
  end record;

  function empty return set is
    s : set;

  begin
    return s;
  end empty;

  procedure empty(ss : out set) is
  begin
```

```
  ss := empty;
end empty;

function add(x : t; s : set) return set is
  ss : set;

begin
  if is_in(x, s) then
    return s;

  else
    ss.p := new set_record'(value => x, rest => s);
    return ss;
  end if;
end add;

procedure add(x : in t; s : in set; ss : out set) is
begin
  ss := add(x, s);
end add;

function remove(x : t; s : set) return set is
  ss : set := empty;

begin    -- begin generator loop
  declare
    exit_from_generator_loop : exception;
    procedure generator_loop_body(y : t) is
    begin

     if not (t_equal(x, y)) then
       ss := add(y, ss);
     end if;
    end generator_loop_body;
    procedure execute_generator_loop is new scan(generator_loop_body);
  begin
    execute_generator_loop(s);
  exception
    when exit_from_generator_loop =>
     null;
  end;                             -- of generator loop
  return ss;
end remove;

procedure remove(x : in t; s : in set; ss : out set) is
begin
  ss := remove(x, s);
end remove;

function is_in(x : t; s : set) return boolean is
begin    -- begin generator loop
  declare
    generator_loop_return_value : boolean;
    return_from_generator_loop  : exception;
    exit_from_generator_loop    : exception;
    procedure generator_loop_body(y : t) is
    begin

     if t_equal(x, y) then
       generator_loop_return_value := true;
       raise return_from_generator_loop;
     end if;
    end generator_loop_body;
```

```
    procedure execute_generator_loop is new scan(generator_loop_body);
  begin
    execute_generator_loop(s);
  exception
    when exit_from_generator_loop =>
     null;
    when return_from_generator_loop =>
     return generator_loop_return_value;
  end;      -- of generator loop
  return false;
end is_in;

procedure is_in(x : in t; s : in set; bb : out boolean) is
begin
  bb := is_in(x, s);
end is_in;

function union(s1, s2 : set) return set is
  ss : set := empty;

begin     -- begin generator loop
  declare
    exit_from_generator_loop : exception;
    procedure generator_loop_body(y : t) is
    begin
     ss := add(y, ss);
    end generator_loop_body;
    procedure execute_generator_loop is new scan(generator_loop_body);
  begin
    execute_generator_loop(s1);
  exception
    when exit_from_generator_loop =>
     null;
  end;                                 -- of generator loop
  declare                              -- begin generator loop
    exit_from_generator_loop : exception;
    procedure generator_loop_body(y : t) is
    begin
     ss := add(y, ss);
    end generator_loop_body;
    procedure execute_generator_loop is new scan(generator_loop_body);
  begin
    execute_generator_loop(s2);
  exception
    when exit_from_generator_loop =>
     null;
  end;     -- of generator loop
  return ss;
end union;

procedure union(s1, s2 : in set; ss : out set) is
begin
  ss := union(s1, s2);
end union;

function difference(s1, s2 : set) return set is
  ss : set := empty;

begin     -- begin generator loop
  declare
    exit_from_generator_loop : exception;
    procedure generator_loop_body(y : t) is
    begin
```

```
      if not is_in(y, s2) then
        ss := add(y, ss);
      end if;
     end generator_loop_body;
     procedure execute_generator_loop is new scan(generator_loop_body);
  begin
    execute_generator_loop(s1);
  exception
    when exit_from_generator_loop =>
     null;
  end;                                    -- of generator loop
  return ss;
end difference;

procedure difference(s1, s2 : in set; ss : out set) is
begin
  ss := difference(s1, s2);
end difference;

function intersection(s1, s2 : set) return set is
  ss : set := empty;

begin    -- begin generator loop
  declare
    exit_from_generator_loop : exception;
    procedure generator_loop_body(y : t) is
    begin

     if is_in(y, s2) then
       ss := add(y, ss);
     end if;
    end generator_loop_body;
    procedure execute_generator_loop is new scan(generator_loop_body);
  begin
    execute_generator_loop(s1);
  exception
    when exit_from_generator_loop =>
     null;
  end;                                    -- of generator loop
  return ss;
end intersection;

procedure intersection(s1, s2 : in set; ss : out set) is
begin
  ss := intersection(s1, s2);
end intersection;

function choose(s : set) return t is
begin
  if size(s) > 0 then
    return s.p.value;

  else
    raise empty_set;
  end if;
end choose;

procedure choose(s : in set; tt : out t) is
begin
  tt := choose(s);
end choose;
```

```
  function size(s : set) return natural is
    k : natural := 0;

  begin    -- begin generator loop
    declare
      exit_from_generator_loop : exception;
      procedure generator_loop_body(y : t) is
      begin
       k := k + 1;
      end generator_loop_body;
      procedure execute_generator_loop is new scan(generator_loop_body);
    begin
      execute_generator_loop(s);
    exception
      when exit_from_generator_loop =>
       null;
    end;                              -- of generator loop
    return k;
  end size;

  procedure size(s : in set; nn : out natural) is
  begin
    nn := size(s);
  end size;

  function equal(s1, s2 : set) return boolean is

  begin
    return subset(s1, s2) and then subset(s2, s1);
  end equal;

  procedure equal(s1, s2 : in set; bb : out boolean) is
  begin
    bb := equal(s1, s2);
  end equal;

function subset(s1, s2 : set) return boolean is
  begin    -- begin generator loop
    declare
      generator_loop_return_value : boolean;
      return_from_generator_loop  : exception;
      exit_from_generator_loop    : exception;
      procedure generator_loop_body(y : t) is
      begin

       if not (is_in(y, s2)) then
         generator_loop_return_value := false;
         raise return_from_generator_loop;
       end if;
      end generator_loop_body;
      procedure execute_generator_loop is new scan(generator_loop_body);
    begin
      execute_generator_loop(s1);
    exception
      when exit_from_generator_loop =>
       null;
      when return_from_generator_loop =>
       return generator_loop_return_value;
    end;     -- of generator loop
    return true;
  end subset;

  procedure subset(s1, s2 : in set; bb : out boolean) is
```

```
begin
  bb := subset(s1, s2);
end subset;

function interval(x1, x2 : in t) return set is
  ss : set := empty;
  y  : t   := x1;

begin
  while not (x2 < y) loop
    ss := add(y, ss);
    y := successor(y);
  end loop;
  return ss;
end interval;

function apply(s : set) return st is
  ss : st := empty;

begin    -- begin generator loop
  declare
    exit_from_generator_loop : exception;
    procedure generator_loop_body(y : t) is
    begin
     ss := add(f(y), ss);
    end generator_loop_body;
    procedure execute_generator_loop is new scan(generator_loop_body);
  begin
    execute_generator_loop(s);
  exception
    when exit_from_generator_loop =>
     null;
  end;                              -- of generator loop
  return ss;
end apply;

function reduce(s : set) return t is
  x : t := identity;

begin    -- begin generator loop
  declare
    exit_from_generator_loop : exception;
    procedure generator_loop_body(y : t) is
    begin
     x := f(y, x);
    end generator_loop_body;
    procedure execute_generator_loop is new scan(generator_loop_body);
  begin
    execute_generator_loop(s);
  exception
    when exit_from_generator_loop =>
     null;
  end;                              -- of generator loop
  return x;
end reduce;

function reduce1(s : set) return t is
  x : t;
  i : natural := 1;

begin
  if size(s) = 0 then
    raise empty_reduction_undefined;
```

```
    end if;
    declare     -- begin generator loop
      exit_from_generator_loop : exception;
      procedure generator_loop_body(y : t) is
      begin

       if i = 1 then
         x := y;

       else
         x := f(y, x);
       end if;
       i := i + 1;
      end generator_loop_body;
      procedure execute_generator_loop is new scan(generator_loop_body);
    begin
      execute_generator_loop(s);
    exception
      when exit_from_generator_loop =>
       null;
    end;     -- of generator loop
    return x;
  end reduce1;

  procedure scan(s : set) is
    ss : set := s;

  begin
    while ss.p /= null loop
      generate(ss.p.value);
      ss := ss.p.rest;
    end loop;
  end scan;

function generic_input return set is
    x  : t;
    ss : set := empty;

  begin
    if token /= '{' then
      raise data_error;
    end if;
    skip_char;
    while token /= '}' loop
      x := input;
      ss := add(x, ss);

      if token = ',' then
       skip_char;

      elsif token /= '}' then
       raise data_error;
      end if;
    end loop;
    skip_char;
    return ss;
  exception
    when others =>
      raise data_error;
  end generic_input;

  function generic_file_input(file : in file_type) return set is
    function get_set is new generic_input;
```

```
    s : set;

  begin
    set_input(file);
    s := get_set;
    set_input(standard_input);
    return s;
  end generic_file_input;

  procedure generic_put(item : in set) is
    i : natural := 1;

  begin
    put(ascii.l_brace);
    declare    -- begin generator loop
      return_from_generator_loop : exception;
      exit_from_generator_loop   : exception;
      procedure generator_loop_body(y : t) is
      begin

       if i > 1 then
         put(", ");
       end if;
       put(y);
       i := i + 1;
      end generator_loop_body;
      procedure execute_generator_loop is new scan(generator_loop_body);
    begin
      execute_generator_loop(item);
    exception
      when exit_from_generator_loop =>
       null;
    end;                                    -- of generator loop
    put(ascii.r_brace);
  end generic_put;

  procedure generic_file_put(file : in file_type; item : in set) is
    procedure put_set is new generic_put;

  begin
    set_output(file);
    put_set(item);
    set_output(standard_output);
  end generic_file_put;
end set_pkg;
```

77. SIMULATION_OBJECT_PKG.ADS

```
-- ------------------------------------------------------------------------
-- File Name:    Simulation_Object_Pkg.Ads
-- Discription:  This package describes the basis for the Simulation Hierarchy
-- ------------------------------------------------------------------------
WITH game_time_type_pkg;  USE game_time_type_pkg;
WITH Location_Type_Pkg;   USE Location_Type_Pkg;

PACKAGE Simulation_Object_Pkg IS

   TYPE Simulation_Object IS ABSTRACT TAGGED PRIVATE;-- Basis of Simulation
Hierarchy
   TYPE Simulation_Object_Ptr IS ACCESS ALL Simulation_Object'Class;


   -- ---------------------------------------------------------------------
```

```
-- PROCEDURE:     Move_Update_Obj
-- PRE:           Obj is of type Simulation_Object and exists
--                Time contains data (value is not never)
-- POST:          Updates Object's location.  Time represents when to
--                reschedule.
-- ---------------------------------------------------------------------
PROCEDURE Move_Update_Obj(Obj   : IN OUT Simulation_Object;
                          Time : IN OUT game_time_type);


-- ---------------------------------------------------------------------
-- FUNCTION:      Can_move
-- ---------------------------------------------------------------------
FUNCTION Can_move(Obj : Simulation_Object) RETURN boolean;

-- ---------------------------------------------------------------------
-- FUNCTION:      Copy_Obj
-- PRE:           Obj is of type Simulation_Object and exists
-- Return:        Makes a copy of the obj and returns a pointer to the new
--                obj
-- ---------------------------------------------------------------------
FUNCTION Copy_Obj(Obj : Simulation_Object) RETURN Simulation_Object_Ptr;


-- ---------------------------------------------------------------------
-- FUNCTION:      Get_Is_Scheduled
-- PRE:           Obj is of type Simulation_Object and exists
-- Return:        Returns the value of Is_Scheduled which is a boolean type
-- ---------------------------------------------------------------------
FUNCTION Get_Is_Scheduled(Obj : Simulation_Object'Class) RETURN Boolean;


-- ---------------------------------------------------------------------
-- PROCEDURE:     Set_Is_Scheduled
-- PRE:           Obj is of type Simulation_Object and exists
-- POST:          Assigns Value to Is_Scheduled
-- ---------------------------------------------------------------------
PROCEDURE Set_Is_Scheduled(Obj    : IN OUT Simulation_Object'Class;
                           Value :        Boolean);


-- ---------------------------------------------------------------------
-- FUNCTION:      Get_Destination
-- PRE:           Obj is of type Simulation_Object and exists
-- Return:        Returns the destination
-- ---------------------------------------------------------------------
FUNCTION Get_Destination(Obj : Simulation_Object'Class)
                                                  RETURN Location_Type;

-- ---------------------------------------------------------------------
-- PROCEDURE:   Set_Destination
-- PRE:           Obj is of type Simulation_Object and exists
-- POST:          Assigns Value to the Destination
-- ---------------------------------------------------------------------
PROCEDURE Set_Destination(Obj : in out Simulation_Object'Class;
                                                   Value: in Location_Type);


-- ---------------------------------------------------------------------
-- FUNCTION:      Get_Location
-- PRE:           Obj is of type Simulation_Object and exists
-- Return:        Returns the location
-- ---------------------------------------------------------------------
```

```
   FUNCTION Get_Location(Obj : Simulation_Object'Class) RETURN Location_Type;


   -- ----------------------------------------------------------------------
   -- FUNCTION:     Get_Fuel_Used
   -- PRE:          Obj is of type Simulation_Object and exists
   -- Return:       Returns the float
   -- ----------------------------------------------------------------------
   FUNCTION Get_Fuel_Used(Obj : Simulation_Object) RETURN Float;


PRIVATE

   TYPE Simulation_Object IS TAGGED RECORD
      Is_Scheduled       : Boolean:=False;
      Name               : String(1..20);
      Graphic_Symbol     : Natural;
      Force              : Natural;  --IE 1..6
      Move_Period        : Integer;
      Active             : Boolean;  --True = active part of sim, ie alive
      Location           : Location_Type;
      Destination        : Location_Type; --Could be a sequence of
                                          --Location_Types
      Speed              : Float; --In M/sec
      Max_Speed          : Float; --In M/sec
   END RECORD;

END Simulation_Object_Pkg;
```

78. SIMULATION_OBJECT_PKG.ADB

```
-- -------------------------------------------------------------------------
-- File Name:    Simulation_Object_Pkg.Adb
-- -------------------------------------------------------------------------

PACKAGE BODY Simulation_Object_Pkg IS

   -- ----------------------------------------------------------------------
   -- PROCEDURE:    Move_Update_Obj
   -- ----------------------------------------------------------------------
   PROCEDURE Move_Update_Obj(Obj  : IN OUT Simulation_Object;
                             Time : IN OUT game_time_type) IS
         Time_Elapsed : Float;  -- In seconds
         Distance     : Float;    -- In meters
         Displacement : Location_Type;
         Velocity     : Location_Type;
      BEGIN
         -- Stop motion if the object cannot move.
         IF not Can_move(Simulation_Object'Class(Obj)) THEN
            Obj.Speed := 0.0;
            Obj.Is_Scheduled := false;
            Time := never; -- Do not reschedule a move event for this object
            return;
         END IF;

         -- How far are we
         Time_Elapsed := Float(Obj.Move_Period);
         Displacement := Obj.Destination - Obj.Location;
         Distance := Length(Displacement);

         -- Set the speed
         -- Future versions will take terrain and weather into account here.
```

```
      IF Distance > Obj.Max_Speed * Time_Elapsed
      THEN Obj.Speed := Obj.Max_Speed;
      ELSE Obj.Speed := Distance/Time_Elapsed;
      END IF;

      -- Move and schedule the next move.
      Velocity := (Obj.Speed/Distance) * Displacement;
      Obj.Location := Obj.Location + (Time_Elapsed * Velocity);
      Time := Time + Obj.Move_Period; --Schedules next event in
                                      --Move_Period seconds

      -- Note: the above code works without modification
      -- for both two and three dimensions.
   END Move_Update_Obj;

-- ---------------------------------------------------------------------
-- FUNCTION:     Can_move
-- ---------------------------------------------------------------------
FUNCTION Can_move(Obj : Simulation_Object) RETURN boolean IS
     Min_Distance : Constant Float := 10.0;
     Distance : Float;
   BEGIN
     Distance := length(Obj.Destination - Obj.Location);
     RETURN Obj.Active -- must be alive to move
            and then Distance > Min_Distance;
                     -- must not already be at the planned destination
   END;

-- ---------------------------------------------------------------------
-- FUNCTION: Copy_Obj
-- ---------------------------------------------------------------------
FUNCTION Copy_Obj(Obj : Simulation_Object) RETURN Simulation_Object_Ptr IS
   BEGIN
      RETURN NULL;  --All are dispatched to leaves of hierarchy
   END Copy_Obj;


-- ---------------------------------------------------------------------
-- FUNCTION: Get_Is_Scheduled
-- ---------------------------------------------------------------------
FUNCTION Get_Is_Scheduled(Obj : Simulation_Object'Class) RETURN Boolean IS
   BEGIN
      RETURN Obj.Is_Scheduled;
   END Get_Is_Scheduled;


-- ---------------------------------------------------------------------
-- PROCEDURE:  Set_Is_Scheduled
-- ---------------------------------------------------------------------
PROCEDURE Set_Is_Scheduled(Obj   : IN OUT Simulation_Object'Class;
                           Value :        Boolean) IS
   BEGIN
      Obj.Is_Scheduled := Value;
   END Set_Is_Scheduled;


-- ---------------------------------------------------------------------
-- FUNCTION:  Get_Destination
-- ---------------------------------------------------------------------
FUNCTION Get_Destination(Obj : Simulation_Object'Class)
                                          RETURN Location_Type IS
   BEGIN
      RETURN Obj.Destination;
```

```
   END Get_Destination;

-- ------------------------------------------------------------------------
-- PROCEDURE:  Set_Destination
-- ------------------------------------------------------------------------
PROCEDURE Set_Destination(Obj : in out Simulation_Object'Class;
                                          Value: in Location_Type) IS
   BEGIN
      Obj.Destination := Value;
   END Set_Destination;


-- ------------------------------------------------------------------------
-- FUNCTION:  Get_Location
-- ------------------------------------------------------------------------
FUNCTION Get_Location(Obj : Simulation_Object'Class) RETURN Location_Type IS
   BEGIN
      RETURN Obj.Location;
   END Get_Location;


-- ------------------------------------------------------------------------
-- FUNCTION:     Get_Fuel_Used
-- ------------------------------------------------------------------------
FUNCTION Get_Fuel_Used(Obj : Simulation_Object) RETURN Float IS
   BEGIN
      RETURN 0.0;
   END Get_Fuel_Used;

END Simulation_Object_Pkg;
```

79. SIMULATION_OBJECT_PKG-GROUND_OBJECT_PKG.ADS

```
-- ------------------------------------------------------------------------
-- File Name:    Simulation_Object_Pkg.Ground_Object_Pkg.Ads
-- ------------------------------------------------------------------------

PACKAGE Simulation_Object_Pkg.Ground_Object_Pkg IS

   TYPE Ground_Object IS ABSTRACT NEW Simulation_Object WITH PRIVATE;

PRIVATE
   TYPE Ground_Object IS ABSTRACT NEW Simulation_Object WITH NULL RECORD;


END Simulation_Object_Pkg.Ground_Object_Pkg;
```

80. SIMULATION_OBJECT_PKG-GROUND_OBJECT_PKG-TANK_PKG.ADS

```
-- ------------------------------------------------------------------------
-- File Name: Simulation_Object_Pkg.Ground_Object_Pkg.Tank_Pkg.Ads
-- ------------------------------------------------------------------------

PACKAGE Simulation_Object_Pkg.Ground_Object_Pkg.Tank_Pkg IS

   TYPE Tank_Type IS NEW Ground_Object WITH PRIVATE;


   -- ---------------------------------------------------------------------
   -- PROCEDURE:     Move_Update_Obj
```

```
--   Description:  Overloaded Simulaiton_Object's Method to work on Tank
--                 Objects
-- ------------------------------------------------------------------------
PROCEDURE Move_Update_Obj(Obj  : IN OUT Tank_Type;
                          Time : IN OUT game_time_type);

-- ------------------------------------------------------------------------
--   FUNCTION:     Can_move
-- ------------------------------------------------------------------------
FUNCTION Can_move(Obj : Tank_Type) RETURN boolean;


-- ------------------------------------------------------------------------
--   FUNCTION:     Get_Fuel_Used
--   Description:  Overloads the SImulation_Object's Method
-- ------------------------------------------------------------------------
FUNCTION Get_Fuel_Used(Obj : Tank_Type) RETURN Float;

-- ------------------------------------------------------------------------
--   FUNCTION:     Copy_Obj
--   Description:  Overloads the SImulation_Object's Method
-- ------------------------------------------------------------------------
FUNCTION Copy_Obj(Obj : Tank_Type) RETURN Simulation_Object_Ptr;

-- ------------------------------------------------------------------------
--   FUNCTION:     Construct_Obj
--   Description:  Constructs a simulation obj
-- ------------------------------------------------------------------------
FUNCTION Construct_Obj(Scheduled     : Boolean;
                       Name          : String;
                       Symbol        : Natural;
                       Force         : Natural;
                       Move_Period   : Integer;
                       Active        : Boolean;
                       Location_x    : Float;
                       Location_y    : Float;
                       Destination_x : Float;
                       Destination_y : Float;
                       Speed         : Float;
                       Max_Speed     : Float;
                       Fuel          : Float;
                       Consumption   : Float) RETURN Simulation_Object_Ptr;


PRIVATE

   TYPE Tank_Type IS NEW Ground_Object WITH RECORD
      Fuel              : Float; --In Gallons
      Fuel_Consumption  : Float; --Gallons/Second
      Fuel_Used         : Float:= 0.0;
   END RECORD;


END Simulation_Object_Pkg.Ground_Object_Pkg.Tank_Pkg;
```

81. SIMULATION_OBJECT_PKG-GROUND_OBJECT_PKG-TANK_PKG.ADB

```
-- ---------------------------------------------------------------------------
-- File Name: Simulation_Object_Pkg.Ground_Object_Pkg.Tank_Pkg.Adb
-- ---------------------------------------------------------------------------
```

```
PACKAGE BODY Simulation_Object_Pkg.Ground_Object_Pkg.Tank_Pkg IS

   -- ----------------------------------------------------------------------
   -- PROCEDURE:    Move_Update_Obj
   -- ----------------------------------------------------------------------
   PROCEDURE Move_Update_Obj(Obj  : IN OUT Tank_Type;
                             Time : IN OUT game_time_type) IS
         Time_Elapsed : Float;  --In Seconds
      BEGIN
         -- Stop motion if the object cannot move.
         IF not Can_move(Obj) THEN
            Obj.Speed := 0.0;
            Obj.Is_Scheduled := false;
            Time := never; -- Do not reschedule a move event for this object
            return;
         END IF;

         -- Move the tank using the general purpose move method
         -- from the most general superclass.
         Move_Update_Obj(Simulation_Object(Obj), Time);

         -- Now do the fuel consumption bookkeeping.
         Time_Elapsed := Float(Obj.Move_Period);
         Obj.Fuel := Obj.Fuel - (Obj.Fuel_Consumption * Time_Elapsed);
         Obj.Fuel_Used := Obj.Fuel_Used + (Obj.Fuel_Consumption
                                               * Time_Elapsed);
      END;


   -- ----------------------------------------------------------------------
   -- FUNCTION:     Can_move
   -- ----------------------------------------------------------------------
   FUNCTION Can_move(Obj : Tank_Type) RETURN boolean IS
      BEGIN
        RETURN Can_move(Simulation_Object(Obj)) -- must satisfy inherited
                                                -- general constraints
               and then Obj.Fuel > 0.0;         -- must also have fuel to move
      END;


   -- ----------------------------------------------------------------------
   -- FUNCTION:     Get_Fuel_Used
   -- Description:  Overloads the SImulation_Object's Method
   -- ----------------------------------------------------------------------
   FUNCTION Get_Fuel_Used(Obj : Tank_Type) RETURN Float IS
      BEGIN
         RETURN Obj.Fuel_Used;
      END;


   -- ----------------------------------------------------------------------
   -- FUNCTION:     Copy_Obj
   -- ----------------------------------------------------------------------
   FUNCTION Copy_Obj(Obj : Tank_Type) RETURN Simulation_Object_Ptr IS
         Obj_Ptr  : Simulation_Object_Ptr;
      BEGIN
         Obj_Ptr:= NEW Tank_Type'(Is_Scheduled      => Obj.Is_scheduled,
                                  Name              => Obj.Name,
                                  Graphic_Symbol    => Obj.Graphic_Symbol,
                                  Force             => Obj.Force,
                                  Move_Period       => Obj.Move_Period,
                                  Active            => Obj.Active,
```

```
                                    Location          => Obj.Location,
                                    Destination       => Obj.Destination,
                                    Speed             => Obj.Speed,
                                    Max_Speed         => Obj.Max_Speed,
                                    Fuel              => Obj.Fuel,
                                    Fuel_Consumption  => Obj.Fuel_Consumption,
                                    Fuel_Used         => Obj.Fuel_Used);
      RETURN Obj_Ptr;
   END;

   -- ---------------------------------------------------------------------
   -- FUNCTION:     Construct_Obj
   -- Description:  Constructs a simulation obj
   -- ---------------------------------------------------------------------
   FUNCTION Construct_Obj(Scheduled     : Boolean;
                          Name          : String;
                          Symbol        : Natural;
                          Force         : Natural;
                          Move_Period   : Integer;
                          Active        : Boolean;
                          Location_X    : Float;
                          Location_Y    : Float;
                          Destination_X : Float;
                          Destination_Y : Float;
                          Speed         : Float;
                          Max_Speed     : Float;
                          Fuel          : Float;
                          Consumption   : Float)
                                        RETURN Simulation_Object_Ptr IS
      Obj_Ptr      : Simulation_Object_Ptr;
      Location     : Location_Type;
      Destination  : Location_Type;
   BEGIN
      Location.X     := Location_X;
      Location.Y     := Location_Y;
      Destination.X  := Destination_X;
      Destination.Y  := Destination_Y;
      Obj_Ptr:= NEW Tank_Type'(Is_Scheduled      => Scheduled,
                               Name              => Name,
                               Graphic_Symbol    => Symbol,
                               Force             => Force,
                               Move_Period       => Move_Period,
                               Active            => Active,
                               Location          => Location,
                               Destination       => Destination,
                               Speed             => Speed,
                               Max_Speed         => Max_Speed,
                               Fuel              => Fuel,
                               Fuel_Consumption  => Consumption,
                               Fuel_Used         => 0.0);
      RETURN Obj_Ptr;
   END;

END Simulation_Object_Pkg.Ground_Object_Pkg.Tank_Pkg;
```

82. SORTED_LIST_PKG.ADS

```
generic
  type element_type is private;
  with function "<"(x, y: element_type) return boolean;
package sorted_list_pkg is
  type sorted_list is private;
```

```
  function empty return sorted_list;
    -- Returns an empty sorted list.

  function is_empty(s: sorted_list) return boolean;
    -- True if and only if s has no elements.

  procedure add(s: in out sorted_list; x: in element_type);
    -- s := s U {x}.

  procedure get_smallest(s: in out sorted_list; x: out element_type);
    -- sets x to the smallest element of s and removes x from s.
    -- raises no_elements if s is empty.

  no_elements: exception;
private
  type sorted_list_record is
    record
      data: element_type;
      next: sorted_list;
    end record;
    -- The list is kept sorted in increasing order wrt "<".
  type sorted_list is access sorted_list_record;
end sorted_list_pkg;
```

83. SORTED_LIST_PKG.ADB

```
-- generic
--   type element_type is private;
--   with function "<"(x, y: element_type) return boolean;
package body sorted_list_pkg is
  free_list: sorted_list := null;

  procedure free(node: sorted_list) is
  begin
    node.next := free_list;
    free_list := node;
  end free;

  function new_node(x: element_type; s: sorted_list)
          return sorted_list is
    node: sorted_list;
  begin
    if free_list = null then
       return new sorted_list_record'(data => x, next => s);
    else node := free_list;
        free_list := free_list.next;
        node.data := x;
        node.next := s;
          return node;
    end if;
  end new_node;

  function empty return sorted_list is
  begin
    return null;
  end empty;

  function is_empty(s: sorted_list) return boolean is
  begin
    return (s = null);
  end is_empty;
```

```
  procedure add(s: in out sorted_list; x: in element_type) is
  begin
    if is_empty(s) then s := new_node(x, s);
    elsif x < s.data then s := new_node(x, s);
    else add(s.next, x);
    end if;
  end add;

  procedure get_smallest(s: in out sorted_list; x: out element_type) is
    head: sorted_list := s;
  begin
    if is_empty(s) then raise no_elements;
    else x := s.data;
        s := s.next;
        free(head);
    end if;
  end get_smallest;

end sorted_list_pkg;
```

84. STATISTICS_REQUEST_TYPE_PKG.ADS

```
package statistics_request_type_pkg is
  type statistics_request_type is private;

  function on return statistics_request_type;

  function off return statistics_request_type;

private
  type statistics_request_type is new boolean;
end statistics_request_type_pkg;
```

85. STATISTICS_REQUEST_TYPE_PKG.ADB

```
package body statistics_request_type_pkg is

  function on return statistics_request_type is
  begin
    return true;
  end on;

  function off return statistics_request_type is
  begin
    return false;
  end off;
end statistics_request_type_pkg;
```

86. STATISTICS_TYPE_PKG.ADS

```
package statistics_type_pkg is
  type statistics_type is private;

  function convert(x: statistics_type) return float;

  function convert(x: float) return statistics_type;

private
  type statistics_type is new float;
end statistics_type_pkg;
```

87. STATISTICS_TYPE_PKG.ADB

```
package body statistics_type_pkg is
  function convert(x: statistics_type) return float is
  begin
    return float(x);
  end convert;

  function convert(x: float) return statistics_type is
  begin
    return statistics_type(x);
  end convert;
end statistics_type_pkg;
```

88. USER_INTERACTION_TYPE_PKG.ADS

```
package user_interaction_type_pkg is
  type user_interaction_type is private;

  function stop_simulation return user_interaction_type;

private
  type user_interaction_type is new boolean;
end user_interaction_type_pkg;
```

89. USER_INTERACTION_TYPE_PKG.ADB

```
package body user_interaction_type_pkg is

  function stop_simulation return user_interaction_type is
  begin
    return true;
  end stop_simulation;
end user_interaction_type_pkg;
```

90. WARRIOR_GLOBAL.H

```
/* *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.] ***
*/
/* *** File:        global.h *** */
/* *** Generated:   Oct 15 11:20:08 1998 *** */
/* ************************************************************************
 * PURPOSE:
 * This global header file is automatically "#include"d in each panel
 * file.  You can insert references to global variables here.
 *
 * REGENERATED:
 * This file is generated only once.  Therefore, you may modify it without
 * impacting automatic code merge.
 *
 * CHANGE LOG:
 * 15-Oct-98    Initially generated...TAE
 * ************************************************************************
 */

#ifndef I_GLOBAL                          /* prevent double include */
#define I_GLOBAL        0

/*      macros for access to parameter values
 *
 * These macros obtain parameter values given the name of
 * a Vm object and the name string of the parameter.
 * The Vm objects are created by the Initialize_All_Panels
 * function for a resource file.
```

```
 *
 * Reference scalar parameters as follows:
 *
 *      StringParm(myPanelTarget, "s")    -- string pointer
 *      IntParm(myPanelTarget, "i")       -- integer value
 *      RealParm(myPanelTarget, "r")      -- real value
 *
 *  For vector parameters, do the following:
 *
 *      TAEINT  *ival;
 *      ival = &IntParm(myPanelTarget, "i");
 *      printf ("%d %d %d", ival[0], ival[1], ival[2]);
 *
 */


#define StringParm(vmId, name)     (SVAL(*Vm_Find(vmId, name),0))
#define IntParm(vmId, name)        (IVAL(*Vm_Find(vmId, name), 0))
#define RealParm(vmId, name)       (RVAL(*Vm_Find(vmId, name), 0))


/*      Dispatch Table typedef          */

typedef VOID (*FUNCTION_PTR) ();
typedef struct DISPATCH
    {
    TEXT                 *parmName;
    FUNCTION_PTR         eventFunction;
    } Dispatch;

#define EVENT_HANDLER static VOID          /* a flag for documentation */

/*      Display Id for use by direct Xlib calls: */

extern Display   *Default_Display;


/*      Externally define wptEvent so event handlers have access to it */

extern WptEvent  wptEvent;  /* event structure returned by Wpt_NextEvent */


#define SET_APPLICATION_DONE \
    { \
    extern BOOL Application_Done; \
    Application_Done = TRUE; \
    }

#endif

/* Automatic TAE-style indenting for Emacs users */
/* *** Local Variables:                                *** */
/* *** mode:                              c            *** */
/* *** c-indent-level:                    0            *** */
/* *** c-continued-statement-offset:      4            *** */
/* *** c-brace-offset:                    4            *** */
/* *** c-brace-imaginary-offset:          4            *** */
/* *** c-argdecl-indent:                  4            *** */
/* *** c-label-offset:                    -4           *** */
/* *** c-continued-brace-offset:          -4           *** */
/* *** comment-column:                    45           *** */
/* *** comment-multi-line:                nil          *** */
/* *** End:                                            *** */
```

91. WARRIOR_PAN_GUI_3.H

```
/* *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.] ***
*/
/* *** File:        pan_gui_3.h *** */
/* *** Generated:   Oct 15 11:20:08 1998 *** */
/* ************************************************************************
 * PURPOSE:
 * Header file for panel:  gui_3
 *
 * REGENERATED:
 * The following WorkBench operations will cause regeneration of this file:
 *     The panel's name is changed (not title)
 * For panel:  gui_3
 *
 * CHANGE LOG:
 * 15-Oct-98    Initially generated...TAE
 * ************************************************************************
 */

#ifndef I_PAN_gui_3                              /* prevent double include */
#define I_PAN_gui_3      0

/* Vm objects and panel Id.   */
extern Id gui_3Target, gui_3View, gui_3Id;

/* Dispatch table (global for calls to Wpt_NewPanel) */
extern struct DISPATCH gui_3Dispatch[];

/* Initialize gui_3Target and gui_3View */
extern VOID gui_3_Initialize_Panel ();

/* Create this panel and display it on the screen */
extern VOID gui_3_Create_Panel ();

/* Destroy this panel and erase it from the screen */
extern VOID gui_3_Destroy_Panel ();

/* Connect to this panel.  Create it or change its state */
extern VOID gui_3_Connect_Panel ();

/*
    extern VOID warrior_Initialize_All_Panels ();
    extern VOID warrior_Create_Initial_Panels ();
*/

/*# MTS 10-15-98
    added the following procedure declarations
#*/

extern VOID set_user_interaction();
extern VOID set_statistics_request();
extern VOID set_replay_request();
extern VOID set_new_plan();

/*# MTS 10-23-98
    added the following function declarations
#*/

extern VOID set_x();
extern VOID set_y();
```

```
FUNCTION VOID display_fuel_consumption();
FUNCTION VOID display_xloc();
FUNCTION VOID display_yloc();
FUNCTION VOID display_mover();
FUNCTION VOID end_simulation();


#endif

/* Automatic TAE-style indenting for Emacs users */
/* *** Local Variables:                             *** */
/* *** mode:                              c         *** */
/* *** c-indent-level:                    0         *** */
/* *** c-continued-statement-offset:      4         *** */
/* *** c-brace-offset:                    4         *** */
/* *** c-brace-imaginary-offset:          4         *** */
/* *** c-argdecl-indent:                  4         *** */
/* *** c-label-offset:                    -4        *** */
/* *** c-continued-brace-offset:          -4        *** */
/* *** comment-column:                    45        *** */
/* *** comment-multi-line:                nil       *** */
/* *** End:                                         *** */
```

92. WARRIOR_CREAT_INIT.C

```
/* *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.] ***
*/
/* *** File:         warrior_creat_init.c *** */
/* *** Generated:    Oct 15 11:20:08 1998 *** */
/* ************************************************************************
 * PURPOSE:
 * Displays all panels in the initial panel set of this resource file
 *
 * REGENERATED:
 * The following WorkBench operations will cause regeneration of this file:
 *     A panel is added to the initial panel set
 *     A panel is deleted from the initial panel set
 *     An initial panel's name is changed (not title)
 * For the set of initial panels:
 *   gui_3
 *
 * CHANGE LOG:
 * MERGE NOTE: Add Change Log entries BELOW this line.
 * 15-Oct-98    Initially generated...TAE
 * MERGE NOTE: Add Change Log entries ABOVE this line.
 * ************************************************************************
 */
#include        "taeconf.inp"
#include        "wptinc.inp"
#include        "warrior_global.h"                  /* Application globals */

/* One include for each panel in initial panel set */
#include        "warrior_pan_gui_3.h"

/* MERGE NOTE: Add additional includes and functions BELOW this line. */
/* MERGE NOTE: Add additional includes and functions ABOVE this line. */


FUNCTION VOID warrior_Create_Initial_Panels ()
    {
    /* MERGE NOTE: Add additional variables and code BELOW this line. */
    /* MERGE NOTE: Add additional variables and code ABOVE this line. */
```

```
    /* Show panels */

    gui_3_Create_Panel (NULL, WPT_PREFERRED);

    /* MERGE NOTE: Add additional code BELOW this line. */
    /* MERGE NOTE: Add additional code ABOVE this line. */
    }

/* Automatic TAE-style indenting for Emacs users */
/* *** Local Variables:                              *** */
/* *** mode:                              c          *** */
/* *** c-indent-level:                    0          *** */
/* *** c-continued-statement-offset:      4          *** */
/* *** c-brace-offset:                    4          *** */
/* *** c-brace-imaginary-offset:          4          *** */
/* *** c-argdecl-indent:                  4          *** */
/* *** c-label-offset:                    -4         *** */
/* *** c-continued-brace-offset:          -4         *** */
/* *** comment-column:                    45         *** */
/* *** comment-multi-line:                nil        *** */
/* *** End:                                          *** */
```

93. WARRIOR_INIT_PAN.C

```
/* *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.] ***
*/
/* *** File:        warrior_init_pan.c *** */
/* *** Generated:   Oct 15 11:20:08 1998 *** */
/* ************************************************************************
 * PURPOSE:
 * Initialize all panels in the resource file.
 *
 * REGENERATED:
 * The following WorkBench operations will cause regeneration of this file:
 *     A panel is deleted
 *     A new panel is added
 *     A panel's name is changed (not title)
 * For the panels:
 *   gui_3
 *
 * CHANGE LOG:
 * MERGE NOTE: Add Change Log entries BELOW this line.
 * 15-Oct-98    Initially generated...TAE
 * MERGE NOTE: Add Change Log entries ABOVE this line.
 * ************************************************************************
 */

#include        "taeconf.inp"
#include        "wptinc.inp"
#include        "symtab.inc"
#include        "warrior_global.h"                 /* Application globals */

/* One "include" for each panel in resource file */
#include        "warrior_pan_gui_3.h"

/* MERGE NOTE: Add additional includes and functions BELOW this line. */
/* MERGE NOTE: Add additional includes and functions ABOVE this line. */


FUNCTION VOID warrior_Initialize_All_Panels (resfileSpec)
    TEXT        *resfileSpec;
    {
    Id vmCollection;
```

```
    /* MERGE NOTE: Add additional variables and code BELOW this line. */
    /* MERGE NOTE: Add additional variables and code ABOVE this line. */

    /* read resource file */
    vmCollection = Co_New (P_ABORT);
    Co_ReadFile (vmCollection, resfileSpec, P_ABORT);

    /* initialize view and target Vm objects for each panel */
    gui_3_Initialize_Panel (vmCollection);

    /* MERGE NOTE: Add additional code BELOW this line. */
    /* MERGE NOTE: Add additional code ABOVE this line. */
    }

/* Automatic TAE-style indenting for Emacs users */
/* *** Local Variables:                               *** */
/* *** mode:                              c           *** */
/* *** c-indent-level:                    0           *** */
/* *** c-continued-statement-offset:      4           *** */
/* *** c-brace-offset:                    4           *** */
/* *** c-brace-imaginary-offset:          4           *** */
/* *** c-argdecl-indent:                  4           *** */
/* *** c-label-offset:                    -4          *** */
/* *** c-continued-brace-offset:          -4          *** */
/* *** comment-column:                    45          *** */
/* *** comment-multi-line:                nil         *** */
/* *** End:                                           *** */
```

94. WARRIOR_PAN_GUI_3.C

```
/* *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.] ***
*/
/* *** File:        pan_gui_3.c *** */
/* *** Generated:   Nov  2 14:32:41 1998 *** */
/* ************************************************************************
 * PURPOSE:
 * This file encapsulates the TAE Plus panel: gui_3
 * These routines enable panel initialization, creation, and destruction.
 * Access to these routines from other files is enabled by inserting
 * '#include "pan_gui_3.h"'.  For more advanced manipulation of the panel
 * using the TAE routines, the panel's Id, Target, and View are provided.
 *
 * NOTES:
 * For each parameter that you have defined to be "event-generating" in
 * this panel, there is an event handler procedure below.  Each handler
 * has a name that is a concatenation of the parameter name and "_Event".
 * Add application-dependent logic to each event handler.  (As generated
 * by the WorkBench, each event handler simply logs the occurrence of the
 * event.)
 *
 * For best automatic code merging results, you should put as many
 * modifications as possible between the lines of the MERGE NOTE comments.
 * Modifications outside the MERGE NOTEs will often merge correctly, but
 * must sometimes be merged by hand.  If the modifications cannot be
 * automatically merged, a reject file (*.rej) will be generated which
 * will contain your modifications.
 *
 * REGENERATED:
 * The following WorkBench operations will cause regeneration of this file:
 *     The panel's name is changed (not title)
 * For panel:  gui_3
 *
```

```
 * The following WorkBench operations will also cause regeneration:
 *     An item is deleted
 *     A new item is added to this panel
 *     An item's name is changed (not title)
 *     An item's data type is changed
 *     An item's generates events flag is changed
 *     An item's valids changed (if item is type string and connected)
 *     An item's connection information is changed
 * For the panel items:
 *   enter_new_plan,  get_re_30,       get_st_27,       get_user_in_21,
 *   get_x,           get_y
 *
 * CHANGE LOG:
 * MERGE NOTE: Add Change Log entries BELOW this line.
 * 2-Nov-98     Initially generated...TAE
 * MERGE NOTE: Add Change Log entries ABOVE this line.
 * ***************************************************************************
 */

#include        "taeconf.inp"
#include        "wptinc.inp"
#include        "warrior_global.h"                /* Application globals */
#include        "warrior_pan_gui_3.h"

/* One "include" for each connected panel */

/* MERGE NOTE: Add includes, vars, and functions BELOW this line. */
/* MERGE NOTE: Add includes, vars, and functions ABOVE this line. */

Id gui_3Target, gui_3View, gui_3Id;
/* gui_3Dispatch is defined at the end of this file */

/* **************************************************************************
 * Initialize the view and target of this panel.
 */
FUNCTION VOID gui_3_Initialize_Panel (vmCollection)
    Id vmCollection;
    {
    gui_3View = Co_Find (vmCollection, "gui_3_v");
    gui_3Target = Co_Find (vmCollection, "gui_3_t");
    }


/* **************************************************************************
 * Create the panel object and display it on the screen.
 */
FUNCTION VOID gui_3_Create_Panel (relativeWindow, flags)
    Window      relativeWindow;
    COUNT       flags;
    {
    /* MERGE NOTE: Add code BELOW this line for gui_3_Create_Panel. */
    /* MERGE NOTE: Add code ABOVE this line for gui_3_Create_Panel. */

    if (gui_3Id)
        printf ("Panel (gui_3) is already displayed.\n");
    else
        gui_3Id = Wpt_NewPanel (Default_Display, gui_3Target, gui_3View,
                                relativeWindow, gui_3Dispatch, flags);
    }


/* **************************************************************************
 * Erases a panel from the screen and de-allocate the associated panel
```

```
 * object.
 */
FUNCTION VOID gui_3_Destroy_Panel ()
    {
    /* MERGE NOTE: Add code BELOW this line for gui_3_Destroy_Panel. */
    /* MERGE NOTE: Add code ABOVE this line for gui_3_Destroy_Panel. */

    Wpt_PanelErase(gui_3Id);
    gui_3Id=0;
    }


/* ************************************************************************
 * Connect to this panel.  Create it or change its state.
 */
FUNCTION VOID gui_3_Connect_Panel (relativeWindow, flags)
    Window      relativeWindow;
    COUNT       flags;
    {
    /* MERGE NOTE: Add code BELOW this line for gui_3_Connect_Panel. */
    /* MERGE NOTE: Add code ABOVE this line for gui_3_Connect_Panel. */

    if (gui_3Id)
        Wpt_SetPanelState (gui_3Id, flags);
    else
        gui_3_Create_Panel (relativeWindow, flags);
    }


/* ************************************************************************
 * Handle event from parameter:  enter_new_plan
 */

EVENT_HANDLER enter_new_plan_Event (value, count)
    TEXT        *value[];                   /* string pointers */
    FUNINT      count;                      /* num of values   */
    {                                       /* parm: enter_new_plan */
    /* MERGE NOTE: Add code BELOW this line for parm: enter_new_plan. */
    /* MERGE NOTE: Add code ABOVE this line for parm: enter_new_plan. */


    set_new_plan();

/*#
    printf ("Panel gui_3, parm enter_new_plan: value = %s\n",
            count > 0 ? value[0] : "none");
#*/
    }                                       /* parm: enter_new_plan */


/* ************************************************************************
 * Handle event from parameter:  get_re_30
 */
EVENT_HANDLER get_re_30_Event (value, count)
    TEXT        *value[];                   /* string pointers */
    FUNINT      count;                      /* num of values   */
    {                                       /* parm: get_re_30 */
    /* MERGE NOTE: Add code BELOW this line for parm: get_re_30. */
    /* MERGE NOTE: Add code ABOVE this line for parm: get_re_30. */

    set_replay_request();
```

```
/*#
    printf ("Panel gui_3, parm get_re_30: value = %s\n",
            count > 0 ? value[0] : "none");
#*/
    }                                       /* parm: get_re_30 */


/* ************************************************************************
 * Handle event from parameter:  get_st_27
 */
EVENT_HANDLER get_st_27_Event (value, count)
    TEXT        *value[];                   /* string pointers */
    FUNINT      count;                      /* num of values   */
    {                                       /* parm: get_st_27 */
    /* MERGE NOTE: Add code BELOW this line for parm: get_st_27. */
    /* MERGE NOTE: Add code ABOVE this line for parm: get_st_27. */

    set_statistics_request();

/*#
    printf ("Panel gui_3, parm get_st_27: value = %s\n",
            count > 0 ? value[0] : "none");
#*/
    }                                       /* parm: get_st_27 */


/* ************************************************************************
 * Handle event from parameter:  get_user_in_21
 */
EVENT_HANDLER get_user_in_21_Event (value, count)
    TEXT        *value[];                   /* string pointers */
    FUNINT      count;                      /* num of values   */
    {                                       /* parm: get_user_in_21 */
    /* MERGE NOTE: Add code BELOW this line for parm: get_user_in_21. */
    /* MERGE NOTE: Add code ABOVE this line for parm: get_user_in_21. */

    set_user_interaction();

/*#
    printf ("Panel gui_3, parm get_user_in_21: value = %s\n",
            count > 0 ? value[0] : "none");
#*/
    }                                       /* parm: get_user_in_21 */


/* ************************************************************************
 * Handle event from parameter:  get_x
 */
EVENT_HANDLER get_x_Event (value, count)
    TAEFLOAT    value[];                    /* real vector     */
    FUNINT      count;                      /* num of values   */
    {                                       /* parm: get_x */
    /* MERGE NOTE: Add code BELOW this line for parm: get_x. */
    /* MERGE NOTE: Add code ABOVE this line for parm: get_x. */

    set_x((double)value[0]);

/*#
    printf ("Panel gui_3, parm get_x: value = %f\n",
            count > 0 ? value[0] : 0);
#*/
    }                                       /* parm: get_x */
```

```
/* ************************************************************************
 * Handle event from parameter:  get_y
 */
EVENT_HANDLER get_y_Event (value, count)
    TAEFLOAT     value[];                          /* real vector      */
    FUNINT       count;                            /* num of values    */
    {                                              /* parm: get_y */
    /* MERGE NOTE: Add code BELOW this line for parm: get_y. */
    /* MERGE NOTE: Add code ABOVE this line for parm: get_y. */

    set_y((double)value[0]);

/*#
    printf ("Panel gui_3, parm get_y: value = %f\n",
            count > 0 ? value[0] : 0);
#*/
    }                                              /* parm: get_y */


struct DISPATCH gui_3Dispatch[] = {
    {"enter_new_plan", enter_new_plan_Event},
    {"get_re_30", get_re_30_Event},
    {"get_st_27", get_st_27_Event},
    {"get_user_in_21", get_user_in_21_Event},
    {"get_x", get_x_Event},
    {"get_y", get_y_Event},
    {NULL, NULL}                                   /* terminator entry */
};


/* MERGE NOTE: Add additional functions BELOW this line. */
/*# MTS 10-15-98
    added the following routines to display info to gui
#*/
FUNCTION VOID display_fuel_consumption(c)
double c;
{
        Wpt_SetReal(gui_3Id, "display_st_31", (TAEFLOAT)c);

}

FUNCTION VOID display_xloc(x)
double x;
{
        Wpt_SetReal(gui_3Id, "xloc", (TAEFLOAT)x);

}

FUNCTION VOID display_yloc(y)
double y;
{
        Wpt_SetReal(gui_3Id, "yloc", (TAEFLOAT)y);

}

FUNCTION VOID display_mover(x, y)
double x, y;
{
        TAEFLOAT value[2];
        value[0] = (TAEFLOAT)x;
        value[1] = (TAEFLOAT)y;
        Vm_SetReal (gui_3Target, "display_re_37", 2, value, P_UPDATE);
```

```
        Wpt_ParmUpdate (gui_3Id, "display_re_37");
}

FUNCTION VOID end_simulation()
{
       Wpt_PanelErase(gui_3Id);
       Wpt_Finish();
        SET_APPLICATION_DONE;
}


/* MERGE NOTE: Add additional functions ABOVE this line. */

/* Automatic TAE-style indenting for Emacs users */
/* *** Local Variables:                                  *** */
/* *** mode:                              c              *** */
/* *** c-indent-level:                    0              *** */
/* *** c-continued-statement-offset:      4              *** */
/* *** c-brace-offset:                    4              *** */
/* *** c-brace-imaginary-offset:          4              *** */
/* *** c-argdecl-indent:                  4              *** */
/* *** c-label-offset:                    -4             *** */
/* *** c-continued-brace-offset:          -4             *** */
/* *** comment-column:                    45             *** */
/* *** comment-multi-line:                nil            *** */
/* *** End:                                              *** */
```

95. warrior_tae.c

```
/* *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.] ***
*/
/* *** File:        warrior.c *** */
/* *** Generated:   Oct 15 11:20:08 1998 *** */
/* ************************************************************************
 * PURPOSE:
 * This the main program of an application generated by the TAE Plus Code
 * Generator.
 *
 * REGENERATED:
 * This file is generated only once.  Therefore, you may modify it without
 * impacting automatic code merge.
 *
 * NOTES:
 * To turn this into a real application, do the following:
 *
 * 1.  Each panel that has event generating parameters is encapsulated by
 * a separate file, named by concatenating the string "pan_" with the
 * panel name (followed by a ".c").  Each parameter that you have defined
 * to be "event-generating", has an event handler procedure in the
 * appropriate panel file.  Each handler has a name that is a
 * concatenation of the parameter name and the string "_Event".  Add
 * application-dependent logic to each event handler.  (As generated by
 * the WorkBench, each event handler simply logs the occurrence of the
 * event.)
 *
 * 2.  To build the program, type "make".  If the symbols TAEINC, ...,
 * are not defined, the TAE shell (source) scripts $TAE/bin/csh/taesetup
 * will define them.
```

```
 *
 * ADDITIONAL NOTES:
 * 1.  Each event handler has two arguments: (a) the value vector
 * associated with the parameter and (b) the number of components.  Note
 * that for scalar values, we pass the value as if it were a vector with
 * count 1.
 *
 * Though it's unlikely that you are interested in the value of a button
 * event parameter, the values are always passed to the event handler for
 * consistency.
 *
 * 2.  You gain access to non-event parameters by calling the Vm package
 * using the targetId Vm objects that are created in
 * Initialize_All_Panels.  There are macros defined in global.h to assist
 * in accessing values in Vm objects.
 *
 * To access panel Id, target, and view, of other panels, add an
 * "#include" statement for each appropriate panel header file.
 *
 * CHANGE LOG:
 * 15-Oct-98    Initially generated...TAE
 * ***********************************************************************
 */

#include        "taeconf.inp"
#include        "wptinc.inp"
#include        "symtab.inc"
#include        "warrior_global.h"                  /* Application globals */
#include        "warrior_pan_gui_3.h"                  /* Application globals
*/


/*     Globally defined variables */

Display *Default_Display;
WptEvent wptEvent;  /* event structure returned by Wpt_NextEvent */
BOOL     Application_Done;

/*# MTS 10-15-98
    replace main routine by initialize_gui and generated_tae_event_monitor

main (argc, argv)

    FUNINT       argc;
    TEXT         *argv[];


    {

#*/
    CODE         eventType;

    COUNT        termLines, termCols;
    CODE         termType;


    /* PROGRAMMER NOTE:
     * add similar extern's for each resource file in this application
     */

    extern VOID warrior_Initialize_All_Panels ();
    extern VOID warrior_Create_Initial_Panels ();
```

```
    struct DISPATCH              *dp;        /* working dispatch pointer */
    struct VARIABLE              *parmv;     /* pointer to event VARIABLE */


/*# MTS 10-15-98
    add the statement void initialize_gui()
#*/

void initialize_gui()
{
    /* initialize terminal without clearing screen */
    t_pinit (&termLines, &termCols, &termType);

    /* permit upper/lowercase file names */
    f_force_lower (FALSE);

    Default_Display = Wpt_Init (NULL);

    /* PROGRAMMER NOTE:
     * To enable scripting, uncomment the following line.  See the
     * taerecord man page.
     */
    /* Wpt_ScriptInit ("warrior"); */

    /* initialize resource file */
    /* PROGRAMMER NOTE:
     * For each resource file in this application, calls to the appropriate
     * Initialize_All_Panels and Create_Initial_Panels must be added.
     */
    warrior_Initialize_All_Panels ("warrior.res");
    warrior_Create_Initial_Panels ();

/*# MTS 10-15-98
    add the following initialization here
#*/
    Application_Done = FALSE;

}


/*# MTS 10-15-98
    commented out the loop and
    add the statment generated_tae_event_monitor()


    /#* main event loop *#/
    /#* PROGRAMMER NOTE:
     #* use SET_APPLICATION_DONE in "quit" event handler to exit loop.
     #* (SET_APPLICATION_DONE is defined in global.h)
     *#/
    while (!Application_Done)

#*/
void generated_tae_event_monitor()
   {
   if (Wpt_Pending())
        {
        eventType = Wpt_NextEvent (&wptEvent); /* get next WPT event */

        switch (eventType)
            {
        case WPT_PARM_EVENT:
```

```
        /* Event has occurred from a Panel Parm.  Lookup the event
         * in the dispatch table and call the associated event
         * handler function.
         */

        dp = (struct DISPATCH *) wptEvent.p_userContext;
        for (;  (*dp).parmName != NULL;   dp++)
            if (s_equal ((*dp).parmName, wptEvent.parmName))
                {
                parmv = Vm_Find (wptEvent.p_dataVm, wptEvent.parmName);
                (*(*dp).eventFunction)
                    ((*parmv).v_cvp, (*parmv).v_count);
                break;
                }
        break;

    case WPT_FILE_EVENT:

        /* PROGRAMMER NOTE:
         * Add code here to handle file events.
         * Use Wpt_AddEvent and Wpt_RemoveEvent to register and remove
         * event sources.
         */
        printf ("No EVENT_HANDLER for event from external source.\n");
        break;

    case WPT_WINDOW_EVENT:

        /* PROGRAMMER NOTE:
         * Add code here to handle window events.
         * WPT_WINDOW_EVENT can be caused by windows which you directly
         * create with X (not TAE panels), or by user acknowledgement
         * of a Wpt_PanelMessage (therefore no default print statement
         * is generated here).
         */
        break;

    case WPT_TIMEOUT_EVENT:

        /* PROGRAMMER NOTE:
         * Add code here to handle timeout events.
         * Timeout events occur when an application has not received any
         * user input within the interval specified by Wpt_SetTimeOut.
         */
        printf ("No EVENT_HANDLER for timeout event.\n");
        break;

    case WPT_TIMER_EVENT:

        /* PROGRAMMER NOTE:
         * Add code here to handle timer events.
         * Timer events occur on (or after) the interval specified when the
         * event is registered using Wpt_AddTimer.  Use Wpt_RemoveTimer to
         * remove timers.
         */
        printf ("No EVENT_HANDLER for event from timer source.\n");
        break;

    default:

        printf("Unknown WPT Event\n");
        break;
        }
```

```
        }
    else if (Application_Done)
        {


            Wpt_Finish();  /* close down all display connections */

        }

    }                                       /* end main */

/* Automatic TAE-style indenting for Emacs users */
/* *** Local Variables:                              *** */
/* *** mode:                             c           *** */
/* *** c-indent-level:                   0           *** */
/* *** c-continued-statement-offset:     4           *** */
/* *** c-brace-offset:                   4           *** */
/* *** c-brace-imaginary-offset:         4           *** */
/* *** c-argdecl-indent:                 4           *** */
/* *** c-label-offset:                   -4          *** */
/* *** c-continued-brace-offset:         -4          *** */
/* *** comment-column:                   45          *** */
/* *** comment-multi-line:               nil         *** */
/* *** End:                                          *** */
```

LIST OF REFERENCES

1. Reengineering Center, *Perspectives on Legacy Systems Reengineering*, pp. 3-5, Software Engineering Institute, 1995.

2. Steigerwald, R, Luqi, and McDowell, J, *CASE Tool for Reusable Software Component Storage and Retrieval in Rapid Prototyping,* Computer Science Dept., Naval Postgraduate School, Monterey, California.

3. Memorandum from Under Secretary of Defense, For Secretary of the Army, SUBJECT: DoD High Level Architecture (HLA) for Simulations, 10 September 1996.

4. Memorandum from Under Secretary of Defense, For Chairman DOD Executive Council for Modeling and Simulation, SUBJECT: DoD Transition to the High Level Architecture (HLA) for Simulations, 7 April 1998.

5. McGinnis, M., Pearman, G., Jackson, and L., Murphy, W., *Development of a PC-based, HLA Compliant, High-resolution, Constructive Combat Simulation,* April 1998.

6. E-mail from Gerry Frazier, Senior Military Analysts, Modeling and Simulation Operations Support Activity, SUBJECT: Constructive Simulation Models, 5 November 1998.

7. Titan, Inc. Applications Group, *Janus 3.X/UNIX Software Design Manual,* Prepared for: Headquarters TRADOC Analysis Center, Ft. Leavenworth, Kansas, November 1993.

8. Titan, Inc. Applications Group, *Janus 3.X/UNIX Software Programmer's Manual.* Prepared for: Headquarters TRADOC Analysis Center, Ft. Leavenworth, Kansas. November 1993.

9. Titan, Inc. Applications Group, Software Simulation Support Group, *The Janus User's Manual Version 3,* pp. 1-15, February 1990.

10. Arnold, R. S. *Software Reengineering.* IEEE Computer Society Press, 1993.

11. Feiler, Peter, *Reengineering: An Engineering Problem,* pp. 97-105, Component-Based Software Engineering, 1990.

12. Luqi and Ketabchi, M., *A Computer-Aided Prototyping System*, 5(2), pp.66-72, IEEE Software, 1988.

13. Luqi, *System Engineering and Computer-Aided Prototyping*, 6(1), pp. 15-17, Journal of Systems Integration – Special Issue on Computer Aided Prototyping, 1996.

14. Austin, Toni, *Idaho National Engineering Laboratory Software Reengineering Fact Sheets*. [http://www.inel.gov/technology_transfer/fact-htm/fact221.html]. July 1996.

15. Kiebak, A., Lichter, H., Schneider-Hufshchmidt, M., and Heinz, Z., *Prototyping in Industrial Software Projects,* undated.

16. Luqi, *A Graph Model for Software Evolution*, IEEE Trans. On Software Engineering, 16, p. 917-927, 8 August 1990.

17. *CAPS User Interface Manual (Section 2)* from http://wwwcaps.cs.nps.navy.mil/Manuals/User_Interface/section2.html

18. Luqi, Steigerwald, R., Hughes G., Naveda F., and Berzins V., *CASP as a Requirements Engineering Tool,* Proc. 1991 Tri-Ada Conference.

19. Berzins, V., Shultes, B., Williams, J., Saluto, M., *Janus Re-engineering Status Brief,* Computer Science Dept., Naval Postgraduate School, October 1998.

BIBLIOGRAPHY

Arnold, R. S. *Software Reengineering*. IEEE Computer Society Press, 1993.

Austin, Toni, *Idaho National Engineering Laboratory Software Reengineering Fact Sheets*. [http://www.inel.gov/technology_transfer/fact-htm/fact221.html]. July 1996.

Berzins, V., Shultes, B., Williams, J., Saluto, M., *Janus Re-engineering Status Brief*, Computer Science Dept., Naval Postgraduate School, October 1998.

CAPS User Interface Manual (Section 2) from http://wwwcaps.cs.nps.navy.mil/Manuals/User_Interface/section2.html

Chikofsky, E.J. & Cross II, J.H., *Reverse Engineering and Design Recovery: A Taxonomy*, IEEE Software, January 1990.

Douglas, Bruce, P., *Real-Time UML: Developing Efficient Objects for Embedded Systems*, pp. 203-218, Addison Wesley Longman, Inc., 1998.

E-mail from Gerry Frazier, Senior Military Analysts, Modeling and Simulation Operations Support Activity, SUBJECT: Constructive Simulation Models, 5 November 1998.

Feiler, Peter, *Reengineering: An Engineering Problem*, pp. 97-105, Component-Based Software Engineering, 1990.

Kiebak, A., Lichter, H., Schneider-Hufshchmidt, M., and Heinz, Z., *Prototyping in Industrial Software Projects*, undated.

Larman, Craig, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design*, pp.273-279, Prentice-Hall PTR, 1998.

L.R. Larimer, *Building an Object Model of a Legacy Simulation*, MS Thesis, NPS, June 1997.

Luqi, *A Graph Model for Software Evolution*, IEEE Trans. On Software Engineering, 16, p. 917-927, 8 August 1990.

Luqi, Steigerwald, R., Hughes G., Naveda F., and Berzins V., *CASP as a Requirements Engineering Tool*, Proc. 1991 Tri-Ada Conference.

Luqi, *System Engineering and Computer-Aided Prototyping*, 6(1), pp. 15-17, Journal of Systems Integration – Special Issue on Computer Aided Prototyping, 1996.

Luqi and Ketabchi, M., *A Computer-Aided Prototyping System*, 5(2), pp.66-72, IEEE Software, 1988.

Memorandum from Under Secretary of Defense, For Chairman DoD Executive Council for Modeling and Simulation, SUBJECT: DoD Transition to the High Level Architecture (HLA) for Simulations, 7 April 1998.

Memorandum from Under Secretary of Defense, For Secretary of the Army, SUBJECT: DoD High Level Architecture (HLA) for Simulations, 10 September 1996.

McGinnis, M., Pearman, G., Jackson, and L., Murphy, W., *Development of a PC-based, HLA Compliant, High-resolution, Constructive Combat Simulation,* April 1998.

Pimper, J. and Dobbs, L., *Janus Algorithm Document*, Version 4.0, Lawrence Livermore National Laboratory, California, 1988.

Reengineering Center, *Perspectives on Legacy Systems Reengineering*, pp. 3-5, Software Engineering Institute, 1995.

Steigerwald, R, Luqi, and McDowell, J, *CASE Tool for Reusable Software Component Storage and Retrieval in Rapid Prototyping*, Computer Science Dept., Naval Postgraduate School, Monterey, California.

Titan, Inc. Applications Group, *Janus 3.X/UNIX Software Design Manual,* Prepared for: Headquarters TRADOC Analysis Center, Ft. Leavenworth, Kansas, November 1993.

Titan, Inc. Applications Group, *Janus 3.X/UNIX Software Programmer's Manual.* Prepared for: Headquarters TRADOC Analysis Center, Ft. Leavenworth, Kansas. November 1993.

Titan, Inc. Applications Group, *Janus Version 6 Data Base Manager's Manual,* Simulation, Training & Instrumentation Command, Orlando, Florida, 1995.

Titan, Inc. Applications Group, Software Simulation Support Group, *The Janus User's Manual Version 3,* pp. 1-15, February 1990.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Rd., STE 0944
   Ft. Belvoir, Virginia 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Road
   Monterey, California 93943-5101

3. Chairman, Code CS 1
   Naval Postgraduate School
   Monterey, California 93943-5100

4. Dr. Man-Tak Shing, Code CS/SH... 1
   Computer Science Department
   Naval Postgraduate School
   Monterey, California 93943-5100

5. Dr. Valdis Berzins, Code CS/BE 1
   Computer Science Department
   Naval Postgraduate School
   Monterey, California 93943-5100

6. U.S Army Research Office 1
   ATTN: Dr. David Hislop (AMXRO-MCS)
   4300 S. Miami Boulevard
   P.O. Box 12211
   Research Triangle Park, North Carolina 27709-2211

7. TRADOC Analysis Center-Monterey.. 1
   Naval Postgraduate School
   P.O. Box 8692
   Monterey, California 93943-0692

8. Director 1
   U.S. Army TRADOC Analysis Center
   ATTN: ATRC
   Ft. Leavenworth, Kansas 66027-5200

9. Director 1
   U.S. Army TRADOC Analysis Center-WSMR
   ATTN: Dr. Mel Parrish
   White Sands Missile Range, New Mexico 88002-5502

10. MAJ Julian R. Williams, Jr. 2
8432 17TH Street North
St. Petersburg, Florida 33702

11. CPT Michael J. Saluto 2
159 Baker Street
East Peoria, Illinois 61611